Relatório da simulação do conversor Buck- Ana Gabriela.

O conversor Buck é um tipo de conversor utilizado para reduzir a tensão de entrada (Vin) para uma tensão menor na saída (Vout), ele é formado por uma fonte de entrada, uma chave, um indutor, um diodo e um capacitor. Neste relatório será analisada a simulação do conversor buck apresentada na aula de eletrônica de potencia do curso de engenharia elétrica da UNEMAT de Sinop.

O código será separado em 8 passos que serão descritos abaixo:

1° Passo foi a importação das bibliotecas utilizadas para as operações matemáticas e plotagem do gráfico:

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   from scipy.integrate import solve_ivp
4
```

• numpy = usada para cálculos numéricos e manipulação de vetores (np.linspace, np.zeros_like, etc).

• matplotlib.pyplot = usada para plotar gráficos de tensão e corrente.

• solve_ivp = função da biblioteca scipy que resolve equações diferenciais ordinárias (ODEs) numericamente.

2° passo é a definição dos parâmetros do circuito:

```
5   # Parâmetros
6   Vin = 24.0
7   L = 72e-6
8   C = 8.69e-6
9   R = 12 / (100/12)
10  f_sw = 100e3
11  Tsw = 1 / f_sw
12  D = 0.5
13
```

Os parâmetros definidos foram:

- **Vin** = 24 V= tensão de entrada
- **L** = 72 µH= indutor
  Determinado a partir da fórmula do indutor:

$$L = \frac{(V_{in} - V_{out}) \cdot D}{\Delta I_L \cdot f_{sw}}$$

- **C** = 8.69 µF= capacitor
  Determinado a partir da fórmula do Capacitor:

$$C = \frac{\Delta I_L}{8 \cdot \Delta V_C \cdot f_{sw}}$$

- **R** = 1.44 Ω= resistência da carga (calculada p/100W a 12V)
- **f_sw** = 100 kHz= frequência de chaveamento
- **Tsw**= período de chaveamento (1/100kHz = 10 µs)
- **D** = 0.5= duty cycle de 50%
  Definido a partir da formula:

$$D = \frac{V_{out}}{V_{in}}$$

  **o Vout não esta descrito nos parâmetros mas é 12V, foi usado nos cálculos realizados para definir os parâmetros como resistência, duty ciclo e indutor antes da elaboração do código.

  3° Passo é a definição das equações diferenciais do conversor:

```
14  def buck_ode(t, x):
15      iL, vC = x
16      if (t % Tsw) < D*Tsw:
17          vL = Vin - vC
18      else:
19          vL = -vC
20      di_dt = vL / L
21      dv_dt = (iL - vC / R) / C
22      return [di_dt, dv_dt]
23
```

- boost_ode= função que representa as equações diferenciais do circuito.
    - t= tempo atual da simulação.
    - x= vetor de estados do sistema
- (t % Ts) < D*Ts → determina se a chave está ligada (ON) ou desligada (OFF).
- (t % Tsw)= devolve quanto tempo se passou desde o início do ciclo atual, sempre "reseta" para zero a cada período de chaveamento.
- D*Tsw= compara se estamos na fase ON do PWM:
    - Se (t % Tsw) < D*Tsw = estamos no ON
    - Caso contrário= OFF.
- Chave On:
    - $V_L = V_{in} - V_{out}$
- Chave OFF:
    - $V_L = -V_{out}$.
- di = vL / L = derivada da corrente do indutor.
- dv = (iL - vC/R) / C = derivada da tensão no capacitor.
- return [di, dv] = retorna a taxa de variação, que solve_ivp integra numericamente.

4° Passo é a definição do Tempo de simulação:

```
24    t_span = (0, 500e-6)
25    t_eval = np.linspace(0, 500e-6, 20000)
26
```

t_span= intervalo de simulação

t_eval= vetor de pontos onde queremos avaliar a solução.

5° Passo solucionar as equações diferenciais:

```
27    sol = solve_ivp(buck_ode, t_span, [0, 0], t_eval=t_eval, max_step=50e-9)
28
```

solve_ivp= Função que resolve EDO.

- buck_ode= Função que contém o modelo matemático do conversor.

- t_span= Intervalo da simulação
- [0, 0]= condições iniciais
- t_eval=t_eval = Instantes de tempo onde a solução será armazenada
- max_step=Ts/50 força a integradora a usar passos pequenos para capturar cada ciclo do conversor. o Ts é o período de chaveamento (ex: 1/100 kHz = 10 µs) o Ts/50 obriga o integrador a nunca avançar mais do que um passo de 10 µs / 50 = 200 ns.

6° Passo extrair os resultados da simulação:

```
29    t = sol.t
30    vC = sol.y[1]
31
```

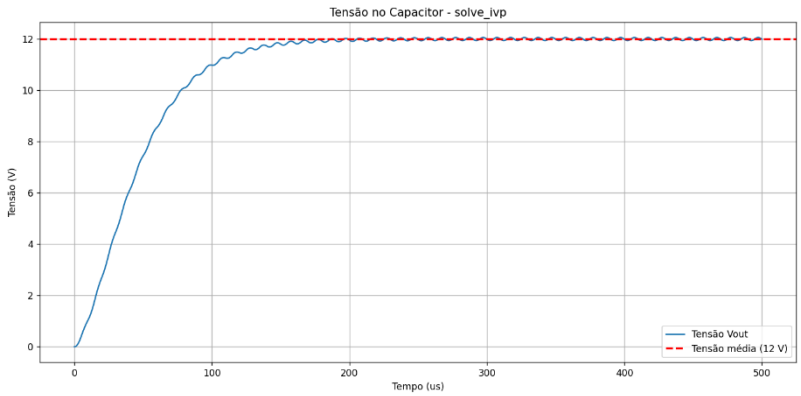
t = vetor de tempo da simulação

vC= variável da tensão no capacitor

7° Passo Plotagem do gráfico:

```
32  # Plot apenas da tensão
33  plt.figure(figsize=(12, 6))
34  plt.plot(t*1e6, vC, label="Tensão Vout")
35  plt.axhline(12, color='red', linestyle='--', linewidth=2, label="Tensão média (12 V)")
36  plt.title("Tensão no Capacitor - solve_ivp")
37  plt.ylabel("Tensão (V)")
38  plt.xlabel("Tempo (us)")
39  plt.grid()
40  plt.legend()
41  plt.tight_layout()
42  plt.show()
```

8° Passo analise do resultado:



O gráfico mostra a **tensão de saída (Vout)** ao longo do tempo em no conversor Buck simulado.

Partida da tensão em 0 V pois as condições iniciais foram definidas em 0 (componente [0,0] no solve_ivp), em seguida vemos a subida exponencial da tensão até próximo de 12 V seguida do regime permanente.

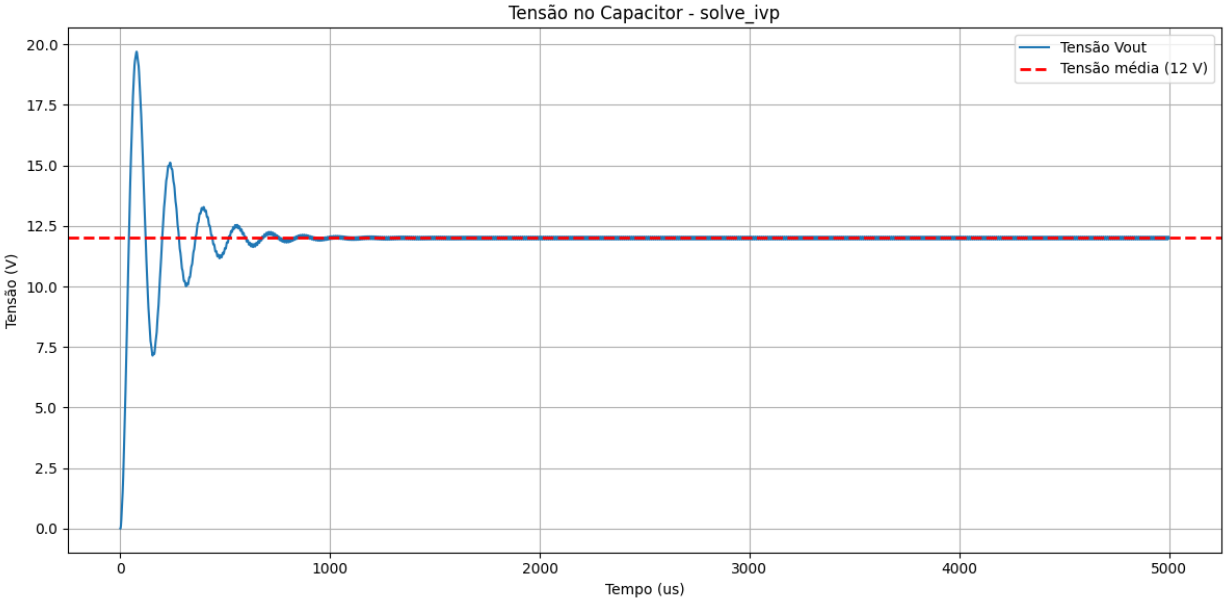
************CORREÇÃO DA SIMULAÇÃO**********

Devido a resistência definida nos parâmetros do circuito estar muito pequena (1,44Ω) ocorre uma diferença na forma de onda de saída gerada pelo código em relação ao esperado, ao aumentar o tamanho da resistência para 10Ω vemos um amortecimento mais adequado ao comportamento do conversor buck:

Figura 1 Aumento do parâmetro de resistência:

```
5    # Parâmetros
6    Vin = 24.0
7    L = 72e-6
8    C = 8.69e-6
9    R = 10
10   f_sw = 100e3
11   Tsw = 1 / f_sw
12   D = 0.5
```

Rodando novamente o código, tendo essa como única modificação obtemos uma nova onda de tensão de saída:


Tensão no Capacitor - solve_ivp
Tensão Vout
Tensão média (12 V)
Tensão (V)
Tempo (us)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1000
2000
3000
4000
5000