EA614 - Análise de Sinais

Atividade 2 - Série de Fourier

Iran Seixas Lopes Neto - RA: 244827
Letícia Lopes Mendes da Silva - RA: 184423

Parte Computacional

Item (a)

Dado o período (T) do enunciado, é possível determinar a frequência fundamental (ω_0) do sinal:

$$T = 7\ s \quad \Rightarrow \quad \omega_0 = \frac{2\pi}{7}\ rad/s$$

Vamos utilizar os coeficientes a_k da série de Fourier da onda $x(t)$:

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t}$$

Da equação de análise, podemos obter os coeficientes:

- $\underline{a_0\ (k=0)}$: Como $e^{-jk\omega_0 t} = 1$ neste caso:

$$a_0 = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)dt$$

$$a_0 = \frac{1}{7}\left[\int_{-2}^{-1} -1dt + \int_{-1}^{0}(t+1)dt + \int_{0}^{1}(t-1)dt + \int_{1}^{2} 1dt\right]$$

$$a_0 = \frac{1}{7}\left[-(-1+2) + (0+\frac{1}{2}) + (-\frac{1}{2}+0) + (2-1)\right] = 0$$

- $\underline{\text{Para } a_k\ (k \neq 0)}$:

$$a_k = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-jk\omega_0 t}\, dt$$

$$a_k = \frac{1}{T}\left[\int_{-2}^{-1}(-1)e^{-jk\omega_0 t}\, dt + \int_{-1}^{0}(t+1)e^{-jk\omega_0 t}\, dt + \int_{0}^{1}(t-1)e^{-jk\omega_0 t}\, dt + \int_{1}^{2} 1\, e^{-jk\omega_0 t}\, dt\right]$$

$$a_k = \frac{1}{T}\left[\frac{je^{jk\omega_0}(e^{jk\omega_0}-1)}{k\omega_0} - \frac{je^{-2jk\omega_0}(e^{jk\omega_0}-1)}{k\omega_0} + \frac{-1+e^{-jk\omega_0}+jk\omega_0}{k^2\omega_0^2} + \frac{1-e^{jk\omega_0}+jk\omega_0}{k^2\omega_0^2}\right]$$

$$a_k = \frac{1}{T}\left[-2j\left(\frac{\sin(k\omega_0) + k\omega_0\cos(k\omega_0)(1 - 2\cos(k\omega_0))}{k^2\omega_0^2}\right)\right]$$

Substituindo os valores $T = 7s$ e $\omega_0 = \frac{2\pi}{7}\, rad/s$, temos:

$$a_k = \frac{-2j}{7}\left[\frac{\sin\left(\frac{2\pi k}{7}\right) + \frac{2\pi k}{7}\cos\left(\frac{2\pi k}{7}\right)\left(1 - 2\cos\left(\frac{2\pi k}{7}\right)\right)}{k^2\left(\frac{2\pi}{7}\right)^2}\right]$$

$$a_k = \frac{7\left[\sin\left(\frac{2k\pi}{7}\right) + \frac{2k\pi\cos\left(\frac{2k\pi}{7}\right)}{7}\left(1 - 2\cos\left(\frac{2k\pi}{7}\right)\right)\right]}{2jk^2\pi^2}$$

Assim:

$$\boxed{a_k = \begin{cases} 0, & k = 0 \\ \dfrac{7\sin\left(\frac{2k\pi}{7}\right) + 2k\pi\cos\left(\frac{2k\pi}{7}\right)\left(1 - 2\cos\left(\frac{2k\pi}{7}\right)\right)}{2jk^2\pi^2}, & k \neq 0 \end{cases}}$$

Para resolver os itens abaixo, utilizamos códigos em Python com as bibliotecas e os dados iniciais a seguir:

```
import numpy as np
import matplotlib.pyplot as plt

T = 7
omega_0 = 2 * np.pi / T
a_0 = 0
t = np.linspace(-(T/2), (T/2), 10_000)

def x_original(t):
    x = np.zeros_like(t)
    x[(t >= -2) & (t < -1)] = -1
    x[(t >= -1) & (t < 0)] = t[(t >= -1) & (t < 0)] + 1
    x[(t >= 0) & (t < 1)] = t[(t >= 0) & (t < 1)] - 1
    x[(t >= 1) & (t < 2)] = 1
    return x

x_t = x_original(t)
```

Item (b)

Para aproximar a onda $x(t)$ pela sua série de Fourier truncada:

$$\tilde{x}_N(t) = \sum_{k=-N}^{N} a_k e^{jk\omega_0 t},$$

implementamos a função `x_fourier(N)`, que encontra os coeficientes a_k de $-N$ até N a partir da relação obtida no item (a), calcula a soma e trata o caso em que $k = 0$ para evitar divisões por zero. Aqui, a série é calculada para um período do sinal $x(t)$.

```
def x_fourier(N):
    x = 0
    for k in range(-N, N+1):
        omega = k * omega_0
        if k == 0:
            a_k = a_0
        else:
            a_k = (1/T)*(-2j*(np.sin(omega) + omega*np.cos(omega)*(1 - 2*
                                                  np.cos(omega)))) / (omega
                                                  )**2
        x += a_k*np.exp(1j*omega*t)
    return x

x_1 = x_fourier(1)
x_10 = x_fourier(10)
x_20 = x_fourier(20)
x_50 = x_fourier(50)
```

Assim, obtivemos a série de Fourier para os valores $N = 1, 10, 20, 50$. A representação gráfica destas séries foram obtidas a partir da função `plot(x_N, N)` e as figuras podem ser visualizadas abaixo.

```
def plot(x_N, N):
    plt.plot(t, x_t, color='tab:red', label=r'$x(t)$')
    plt.scatter(x=t, y=x_N.real, s=2, label=rf'$\tilde{{x}}_{{{N}}} (t)$')
    plt.xlabel(r'$t$')
    plt.ylabel(r'$x(t)$')
    plt.grid(True)
    plt.legend()
    plt.show()

plot(x_1, 1)      # N = 1
plot(x_10, 10)    # N = 10
plot(x_20, 20)    # N = 20
plot(x_50, 50)    # N = 50
```

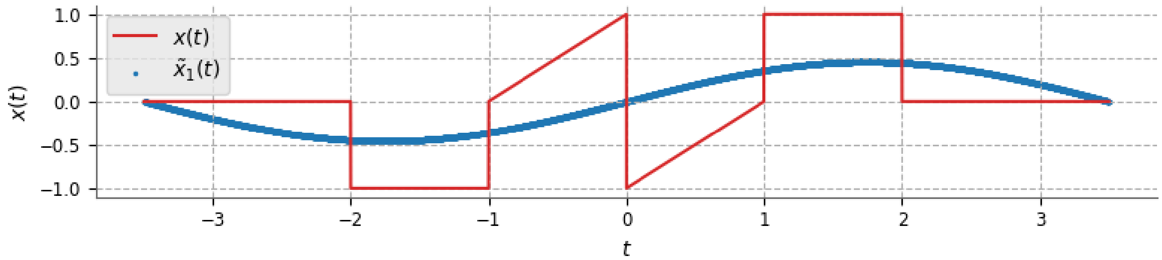

Figura 1: Onda $x(t)$, em vermelho, e sua aproximação $\tilde{x}_N(t)$ para $N = 1$, em azul.

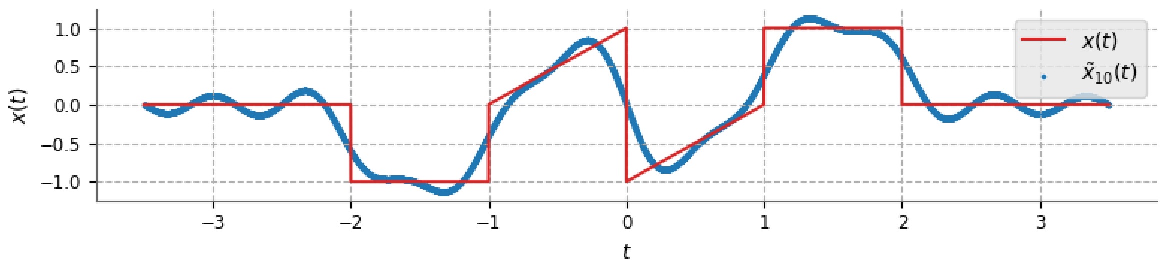

Figura 2: Onda $x(t)$, em vermelho, e sua aproximação $\tilde{x}_N(t)$ para $N = 10$, em azul.

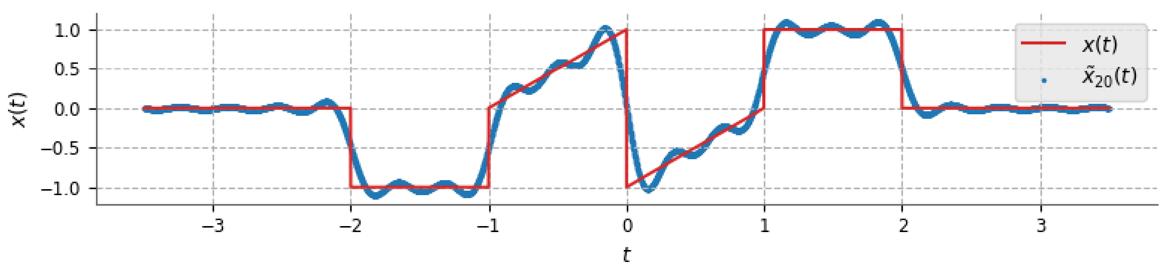

Figura 3: Onda $x(t)$, em vermelho, e sua aproximação $\tilde{x}_N(t)$ para $N = 20$, em azul.

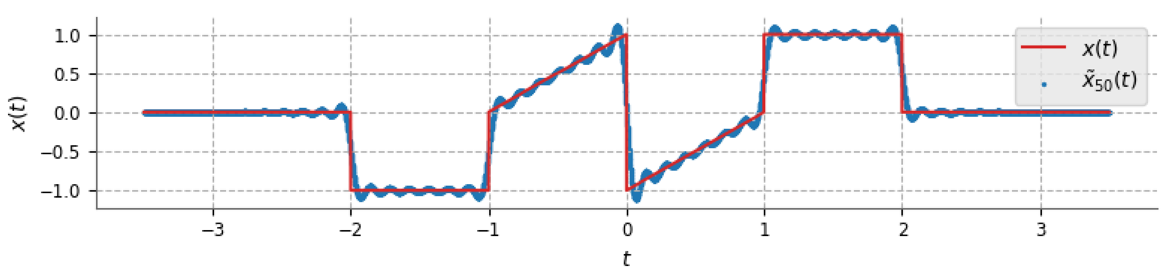

Figura 4: Onda $x(t)$, em vermelho, e sua aproximação $\tilde{x}_N(t)$ para $N = 50$, em azul.

Analisando as séries calculadas, verifica-se que, quanto maior o valor de N, maior é a precisão do sinal em relação a $x(t)$. Isto é, quando $N = 1$, a aproximação é muito grosseira, já que utiliza apenas o termo da frequência fundamental. Já quando $N = 50$, com um número considerável de frequências, a série se aproxima bastante de $x(t)$ até em regiões de descontinuidades, o que verifica a convergência da série de Fourier para o sinal periódico estudado.

No entanto, mesmo para um N elevado, nota-se que ainda há ondulações próximas às regiões de descontinuidade que não convergem. Estes pequenos "saltos" representam o fenômeno de Gibbs, que ocorre ao representar um sinal com descontinuidade pela série de Fourier. À medida que N cresce, estas ondulações ficam cada vez mais próximas à região descontínua, indicando que é necessário escolher um valor para N suficientemente grande para garantir que estes "saltos" sejam os menores possíveis.

Item (c)

Primeiro, usando os sinais de série aproximados $\tilde{x}_N(t)$, calculamos, a partir da função `pot(x_original, x_fourier)`, o quadrado de cada erro em relação a $x(t)$:

$$e^2(t) = (x(t) - \tilde{x}_N(t))^2$$

Depois, calculamos a média desse valor, usando o comando `np.mean()`, para encontrar a potência média do erro. Assim, obtivemos os seguintes valores de potência para cada N:

```python
def pot(x_original, x_fourier):
    err = (x_original - x_fourier)**2
    return np.mean(err) # Aprox discreta da integral

P_1 = pot(x_t, x1)
P_10 = pot(x_t, x10)
P_20 = pot(x_t, x20)
P_50 = pot(x_t, x50)

print(f'Potencia media do erro para N = 1: {P_1}')
print(f'Potencia media do erro para N = 10: {P_10}')
print(f'Potencia media do erro para N = 20: {P_20}')
print(f'Potencia media do erro para N = 50: {P_50}')
```

$$P_N = \begin{cases} 0.2767653611529634 + 0j, & N = 1 \\ 0.04037836598957219 - 7.69955352444065 \times 10^{-20}j, & N = 10 \\ 0.019391980399077604 - 5.604020146655687 \times 10^{-20}j, & N = 20 \\ 0.008216140378160625 - 2.758788135161333 \times 10^{-20}j, & N = 50 \end{cases}$$

É possível notar que a potência tende a 0 conforme N aumenta. Isso acontece pois o sinal $\tilde{x}_N(t)$ se aproxima cada vez mais de $x(t)$ quanto maior o número de iterações (conforme visto no item (b)), diminuindo a energia do erro $e(t)$. Portanto, se o erro diminui, a média do erro quadrado, isto é, a potência, também diminui e, com isso, melhor é a aproximação.

Item (d)

Para calcular o módulo dos coeficientes $|a_k|$, usamos o comando `np.abs()` para obter o valor absoluto de a_k e, então, criamos o gráfico usando o comando `plt.stem()`:

```
N = 50
k = np.linspace(-N, N, 2*N)
omega = k * omega_0
a_k = (1/T)*(-2j*(np.sin(omega) + omega*np.cos(omega)*(1 - 2*np.cos(omega)
                                        ))) / (omega)**2
a_k = np.abs(a_k)

plt.stem(omega, a_k, basefmt=" ")
plt.xlabel(r'$\omega = k \cdot \omega_0$')
plt.ylabel(r'$|a_k|$')
plt.grid(True)
plt.show()
```

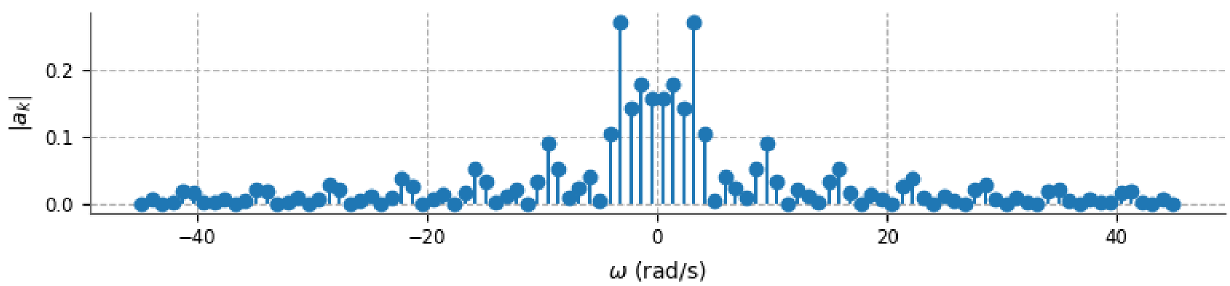

Figura 5: Módulo dos coeficientes $|a_k|$ em função de $\omega = k \cdot \omega_0$.

Do gráfico, observamos que os módulos dos coeficientes têm simetria par, ou seja, $|a_k| = |a_{-k}|$. Isso faz sentido quando analisamos o sinal $x(t)$, que possui simetria ímpar ($x(t) = -x(-t)$). Pela propriedade de reversão no tempo da série de Fourier, temos que, se $x(t) \Leftrightarrow a_k$, então $x(-t) \Leftrightarrow a_{-k}$. Logo, temos que:

$$a_k = -a_{-k} \;\Rightarrow\; |a_k| = |-a_{-k}| = |a_{-k}|$$

Item (e)

Usando os valores de R e C fornecidos, calculamos a frequência de corte ω_c e, então, o valor da resposta em frequência $H(j\omega)$. Depois disso, calculamos o módulo da resposta $|H(j\omega)|$ usando o comando `np.abs()` e a sua fase $\angle H(j\omega)$ com o comando `np.angle()`. Escolhemos um intervalo para ω suficientemente grande para analisar o comportamento de $H(j\omega)$ para pequenas e altas frequências. Com isso, criamos os os gráficos de seu módulo e de sua fase, representando, em vermelho, a frequência de corte $\omega_c = \dfrac{1}{RC}$ do filtro.

```python
# Valores RC
R = 100_000 # 100 kOhms
C = 1e-6    # 1 microF
omega_c = 1 / (R * C) # Freq de corte

def Hjw(omega):
    return 1 / (1 - 1j * (omega_c / omega))

omega = np.linspace(-100, 100)
mod_H = np.abs(Hjw(omega))
angle_H = np.angle(Hjw(omega))

plt.figure(figsize=(8, 4))

plt.subplot(2, 1, 1)
plt.plot(omega, mod_H)
plt.ylabel(r'$|H(j\omega)|$')
plt.grid(True)
plt.axvline(x = omega_c, color = 'tab:red', ls = '--')
plt.axvline(x =-omega_c, color = 'tab:red', ls = '--')

plt.subplot(2, 1, 2)
plt.plot(omega, angle_H)
plt.ylabel(r'$\angle H(j\omega)$')
plt.grid(True)
plt.axvline(x = omega_c, color = 'tab:red', ls = '--')
plt.axvline(x =-omega_c, color = 'tab:red', ls = '--')

plt.xlabel(r'$\omega$ (rad/s)')
plt.show()
```

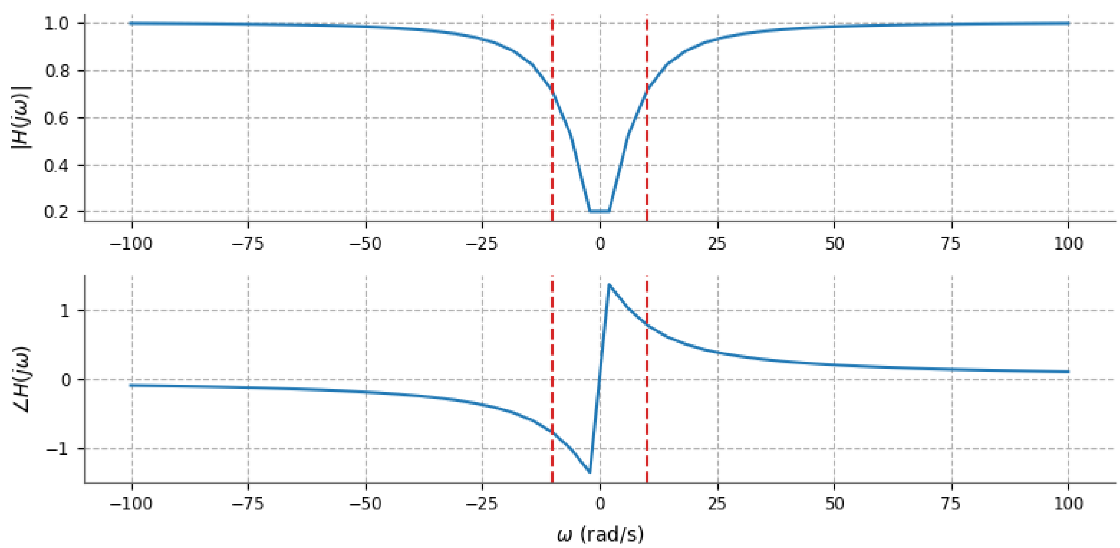

Figura 6: Módulo da resposta em frequência $H(j\omega)$, na parte superior, e sua fase, na parte inferior.

Ao analisar o gráfico $|H(j\omega)|$, nota-se que o módulo se aproxima de 0 quando $\omega \to 0$, e tende a 1 para grandes valores de $|\omega|$. Logo, é possível afirmar que o circuito RC dado atua como um filtro passa-altas (FPA). Além disso, a transição da faixa de passagem e de rejeição não é instantânea, isto é, não ocorre exatamente no valor de frequência de corte $\omega_c = 10$ rad/s, sendo o circuito, então, um FPA não-ideal.

Item (f)

Dado que o sistema em questão é um sistema LIT, valem as propriedades de autofunção e autovalor. Então, a saída $y(t)$ pode ser representada por uma série de Fourier, com coeficientes $c_k = a_k H(jk\omega_0)$:

$$\tilde{y}(t) = \sum_{k=-N}^{N} a_k H(jk\omega_0) e^{jk\omega_0 t},$$

```
def y_output(N):
    y = 0
    for k in range(-N, N+1):
        omega = k * omega_0
        if k == 0:
            a_k = a_0
            H_k = 0
        else:
            a_k = (1/T)*(-2j*(np.sin(omega) + omega*np.cos(omega)*(1 - 2*
                                                np.cos(omega))) / (omega
                                                )**2
            H_k = Hjw(omega)  # Aplicando H(jw) para cada k
        y += a_k*H_k*np.exp(1j*omega*t)
    return y

y_t = y_output(N)

plt.figure(figsize=(8, 4))
plt.plot(t, y_t.real, label=r'Saida $\tilde{y}(t)$ do filtro')
plt.plot(t, x_50.real, label=r'Entrada $\tilde{x}_{50} (t)$', ls = '--')
plt.xlabel('t (s)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.show()
```

Analisando a imagem obtida (Figura 7), nota-se que o sinal aproximado $\tilde{y}(t)$ para a saída do sistema LIT tem o mesmo período do sinal de entrada $\tilde{x}_N$. Logo, suas frequências fundamentais são as mesmas (ω_0). Além disso, é possível observar que a amplitude da saída é acentuada quando há mudanças abruptas no sinal de entrada, como nos instantes $t = -2, -1, 0, 1, 2$, onde há descontinuidade. Isto indica que o filtro está enfatizando as variações rápidas do sinal, característica típica de um filtro passa-alta, coincidindo com a análise feita no item (e).

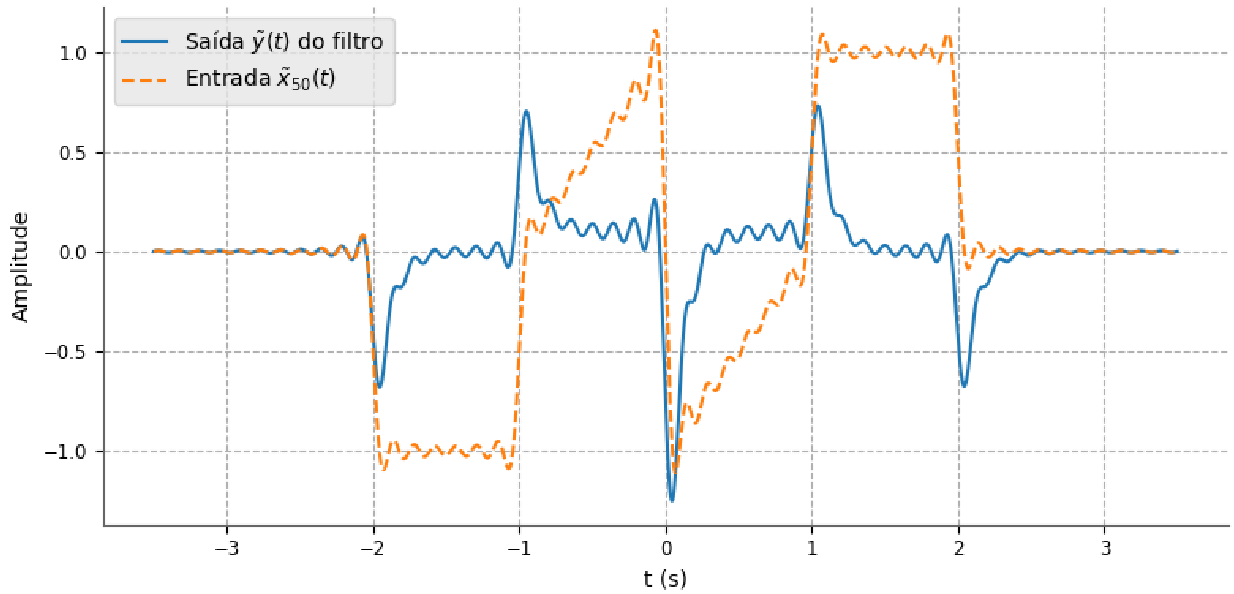

Figura 7: Entrada $\tilde{x}_{50}(t)$, em amarelo, e saída $\tilde{y}(t)$ após passar pelo sistema LIT, em azul.

Item (g)

Comparando a resposta do sistema LIT ao sinal $x(t)$ com a resposta ao sinal $\tilde{x}_{50}(t)$, observamos que ambos tem um formato semelhante. Porém, a resposta $\tilde{y}(t)$ possui vários "saltos", o que pode ser explicado pelo fenômeno de Gibbs, já que esse sinal também é uma série de Fourier. Além disso, $\tilde{y}(t)$ possui pontos de mínimo e máximo menores em módulo se comparado a $y(t)$, sendo resultado da multiplicação da autofunção $e^{jk\omega_0 t}$ com o autovalor $H(jk\omega_0)$. Isso ocorre devido às limitações durante a obtenção de $\tilde{y}(t)$, pois a aproximação utiliza frequências somente até a 50ª harmônica.