UNIVERSIDADE FEDERAL DO ABC- UFABC

Prova 1 - Transformadas em Sinais e Sistemas Lineares

Gabriel de Oliveira Souza - RA.:11201811094

SANTO ANDRE - SP
Julho 2021

P1 - TSSL

Gabriel de Oliveira Souza

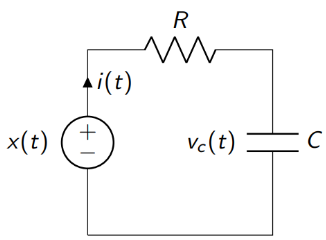
Julho 2021

1 Enunciado do problema

Primeira avaliação

Enunciado do problema

Para o circuito mostrado



Considere o sinal de tensão no capacitor $v_c(t)$ como a saída do circuito.
Sabendo que

$$v_c(t) = \frac{1}{C}\int_0^t i(\tau)d\tau,$$

o modelo matemático é

$$(D + 1/RC)v_c(t) = (1/RC)x(t)$$

com $R = 0.8\Omega$ e $C = 0.1F$.

1) Calcule $V_c(t)$ para o sinal periódico quadrado mostrado com $T_0 = 4$ e $T_s = 1$, considere nula a carga inicial do capacitor, isto é, $V_c(0) = 0$

2) Obtenha os gráficos dos sinais $V(t)$, $I(t)$ e $V_c(t)$

3) Obtenha os espectros de amplitude e fase dos sinais obtidos.

4) Repita o exercício anterior para o sinal de entrada com $A = 2$, $T_0 = 5$, $T_H = 2$ e $T_L = 3$

2 Resolução do problema 1)

O primeiro passo para obtenção do sinal de saída $V_c(t)$ é a modelagem do sinal de entrada x(t) a partir da série de Fourier contínua no tempo.

Para modelagem e cálculo da série, foi adotada como ferramenta, a linguagem de programação Python 3.8, juntamente com as bibliotecas Numpy, Sympy e Matplotlib para manipulação numérica, simbólica e visualização dos dados, respectivamente.

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as mpl
from IPython.display import display, Math
from sympy import symbols
```

Particularmente, x(t) se trata de um sinal periódico quadrado e pode ser facilmente modelado pela série de Fourier tomando apenas um período do sinal, neste caso escolhemos o intervalo compreendido entre $-T_0$ e T_0, onde o comportamento do sinal pode ser descrito pela seguinte função.

$$x(t) = 1, -Ts \leq x \leq Ts$$
$$x(t) = 0, otherwise$$

Para obter a série de Fourier correspondendo ao sinal, primeiro temos de calcular os Harmônicos da série (X_k), que podem ser obtidos através da equação (1):

$$X_k = \frac{1}{T_0} \int_{T_0} x(t) e^{-jk\omega_0 t} dt \tag{1}$$

Substituindo os valores do intervalo de integração escolhido, podemos definir a integral dos harmônicos da série pela expressão (2):

$$0.25 \int\limits_{-1}^{1} e^{-\frac{i\pi kt}{2}}\, dt \tag{2}$$

Ao resolver esta integral, chegamos que os harmônicos da série de Fourier para o sinal x(t) são dados por:

$$\begin{cases} \frac{1.0 \sin\left(\frac{\pi k}{2}\right)}{\pi k} & \text{for } k > -\infty \wedge k < \infty \wedge k \neq 0 \\ 0.5 & \text{otherwise} \end{cases} \tag{3}$$

Através do código Python a seguir, foi possível definir e calcular a integral (2), obtendo o resultado (3)

```
t , j , k , u, w = symbols('t j k u w')
T0 = 4
Ts = 1
w0 = 2*sp.pi/T0
x = 1
auto = sp.exp(-sp.I*k*w0*t)
Xk = (1/T0)*sp.Integral(x*auto,(t,-Ts,Ts))
Xks = sp.simplify(sp.combsimp(Xk.doit()))
Xks
```

Agora com a expressão dos harmônicos bem definida, podemos obter a forma trigonométrica da série de Fourier, calculada através da equação (4).

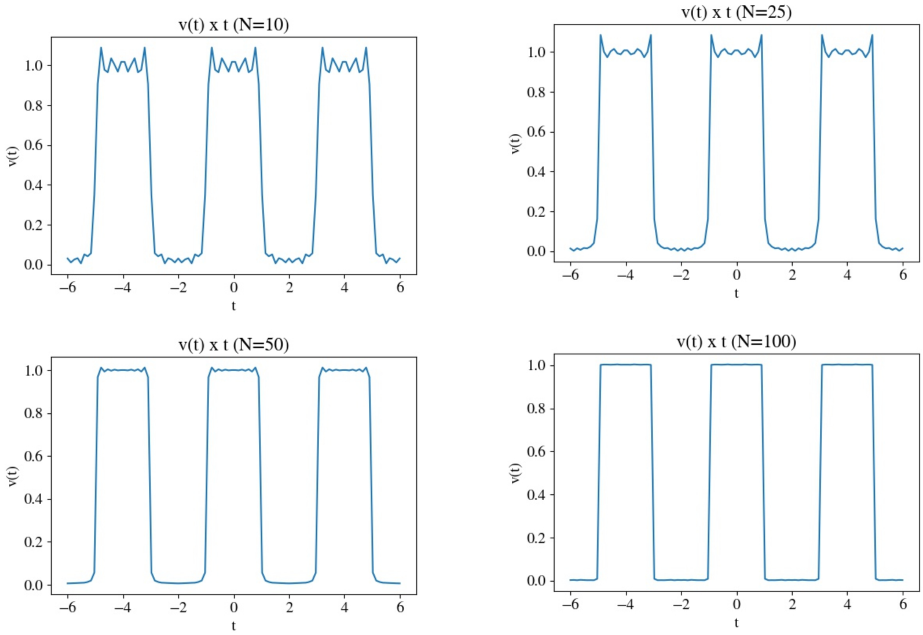
$$x(t) = X_0 + \sum_{k \in \mathbb{N}} \left(2Re(X_k) \cos\left(k\omega_0 t\right) - 2Im(X_k) \operatorname{sen}\left(k\omega_0 t\right)\right) \tag{4}$$

Através do código Python a seguir, foi possível calcular a série de Fourier trigonométrica, como definida em (4):

```
def round_expr(expr, num_digits):
  return expr.xreplace({n : round(n, num_digits)
  for n in expr.atoms(sp.Number)})
# Série de Fourier Trigonométrica
x = sp.Function("x")
x = 0
#X0
x =  Xks.subs(k,0)
#Somatória
for i in range(1,100):
  x += ( 2*sp.re(Xks.subs(k,i))*sp.cos(k*w0*t)
```

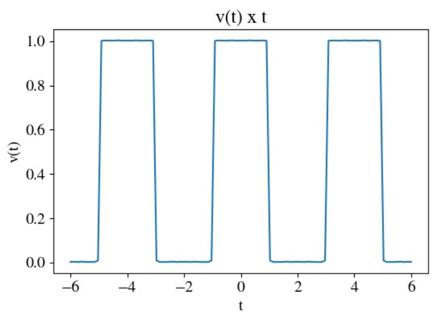
```
  \-2*sp.im(Xks.subs(k,i))*sp.sin(k*w0*t)).subs(k,i)
x = round_expr(x,3)
#Mapeando a funcao do Sympy p/ Numpy
lam_x = sp.lambdify(t, x, modules=['numpy'])
lam_x
```

Plotando o resultado da série trigonométrica de Fourier para o sinal x(t), com N=10, 25, 50 e 100 termos na composição da série, obtemos os seguintes resultados:



Comparativo 1: Convergência Série de Fourier

Para manter maior fidelidade ao sinal original x(t), seguiremos com a série de Fourier trigonométrica de V(t) composta por 100 termos (N=100). Ao plotar esta série, temos o resultado abaixo:



Agora, com o sinal da tensão de entrada x(t) já modelado através da série de Fourier trigonométrica, podemos determinar $V_c(t)$, resolvendo a Equação Diferencial (5)

$$\begin{aligned} (D + 1/RC)V_c(t) = (1/RC)x(t) \\ V_c(0) = 0 \\ \text{com } R = 0.8\Omega \text{ e } C = 0.1F \end{aligned} \tag{5}$$

Para resolver a Equação (5), temos de primeiramente calcular a resposta ao impulso unitário $h(t)$, referente à tensão de sáida $V_c(t)$. O primeiro passo, consiste na obtenção dos termos dos modos característicos $y_n(t)$, através da Equação (6).

$$12.5\,\mathrm{y_n}(t) + \frac{d}{dt}\mathrm{y_n}(t) = 0 \tag{6}$$

Resolvendo a Equação diferecial (6), temos que os TMC, são da forma:

$$\mathrm{y_n}\,(t) = e^{-12.5t}$$

Podemos obter a resposta ao impulso unitário $h(t)$, através da Equação (7)

$$h(t) = b_0\delta(t) + [P(D)y_n(t)]\,u(t) \tag{7}$$

$$\implies h(t) = 12.5\delta\,(t) + 12.5e^{-12.5t}, t \geq 0$$

Através do código Python a seguir, foi possível obter a expressão da reposta ao impulso unitário $h(t)$, seguindo os passos descritos por (5), (6) e (7):

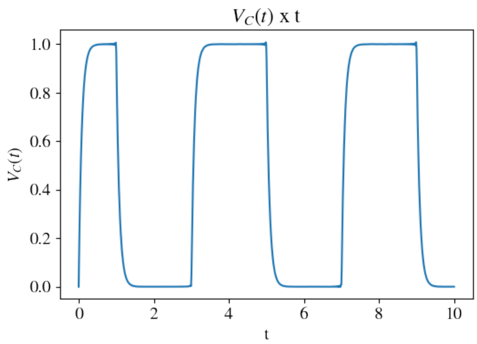
```
y_n = sp.Function('y_n')
h_t = sp.Function('h_t')

#Definindo a equação para encontrarmos y_n(t)
eq = sp.Eq(sp.diff(y_n(t), t,1) + (1/(R*C))*y_n(t),0)
display(eq)

#Resolvendo a equação com as condições iniciais da função de DIRAC

res = sp.dsolve(eq,hint="best")
y_n = res
display(y_n)

#Substituindo as cond iniciais do sistema

const = sp.solve(sp.Eq(y_n.rhs.subs(t,0),1))[0]
display(const)

y_n = y_n.rhs.subs("C1",const)

#Aplicar o operador P(D) no resultado obtido em y_n
h_t =  (1/(R*C))*y_n
display(h_t)
h_t = sp.simplify(sp.combsimp(h_t.doit())) + (1/(R*C))*sp.DiracDelta(t)
display(Math("h_t(t) ={}"+ sp.latex(h_t) + " , t\geq0"))
```

Partindo do resultado de $h(t)$ obtido em (7), foi possível obter $V_c(t)$, através da operação de convolução entre $h(t)$ e $x(t)$, representada pela Equação (8):

$$V_c(t) = (x * h)(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \tag{8}$$

$$\implies \int\limits_0^t \left(12.5e^{12.5\tau - 12.5t} + 12.5\delta\,(-\tau + t)\right) x(\tau)\,d\tau$$

Calculando a integral de convolução (8) no Python, podemos visualizar a curva de $V_c(t)$ X t. Plotando a função resultante, foi obtido o seguinte resultado:

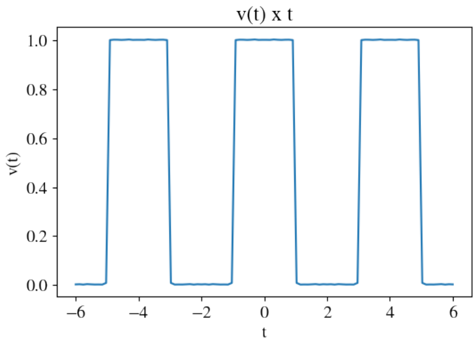


Através do código Python a seguir, foi possível obter a solução da integral de convolução da Equação (8):

```
def resposta_estado_nulo(x_t,h_t):
  display(Math("\n h(t) ={}" + sp.latex(h_t)), Math("x(t) ={}" + sp.latex(x_t)))

  expr = sp.Integral(x_t.subs(t,j)*h_t.subs(t,t-j),(j,0,t))
  display(expr)

  result = expr.doit()
  result = round_expr(result,3)

  display(Math("\ny_h(t) ={}" + sp.latex(result) + " , t\geq0"))
  return result

V_c = sp.Function("V_c")
V_c = resposta_estado_nulo(x,h_t)
```

3 Resolução do problema 2)

Conforme apresentado na Seção 2, o gráfico de V(t) foi obtido a partir da série de Fourier trigonométrica composta por 100 termos (N=100). Podemos observar o plot abaixo:



Código Python utilizado para plot do sinal $V(t)$:

```
lam_x = sp.lambdify(t, x, modules=['numpy'])
x_vals = np.linspace(-T0-Ts-1, T0+Ts+1, 100)
y_vals = (abs(lam_x(x_vals)))
mpl.plot(x_vals, y_vals)
mpl.ylabel("v(t)")
mpl.xlabel("t")
mpl.title("v(t) x t")
mpl.savefig("v_100.jpg")
mpl.show()
```

Analogamente ao procedimento apresentado para obtenção do sinal $V_C(t)$ durante a Seção 2, foi necessário calcular a resposta ao impulso unitário $h(t)$ e realizar a operação de convolução com o sinal de entrada $V(t)$, para obter o gráfico de I(t).

Podemos determinar I(t), resolvendo a Equação Diferencial (9)

$$\begin{aligned} (D + 1/RC)I(t) &= (1/R)Dx(t) \\ I(0) &= 1 \\ R = 0.8\Omega \text{ e } C &= 0.1F \end{aligned} \tag{9}$$

$$12.5\, \mathrm{y_n}(t) + \frac{d}{dt}\mathrm{y_n}(t) = 0 \implies \mathrm{y_n}(t) = C_1 e^{-12.5t} \tag{10}$$

Podemos obter a resposta ao impulso unitário $h(t)$, através da Equação (7)

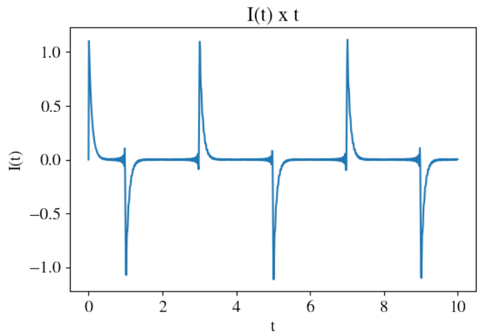
$$\implies h_t(t) = 1.25\delta\left(t\right) - 15.625e^{-12.5t}, t \geq 0 \tag{11}$$

Através do código Python a seguir, foi possível obter a expressão da reposta ao impulso unitário $h(t)$, seguindo os passos descritos por (9), (10) e (11):

```
#Definindo os parâmetros inciais do sistema
R = 0.8
C = 0.1
y_n = sp.Function('y_n',real=True)
h_t = sp.Function('h_t',real=True)

#Definindo a equação para encontrarmos y_n(t)
eq = sp.Eq(sp.diff(y_n(t), t,1) + (1/(R*C))*y_n(t),0)
display(eq)

#Resolvendo a equação com as condições iniciais da função de DIRAC
res = sp.dsolve(eq,hint="best")
y_n = res
display(y_n)

#Substituindo as cond iniciais do sistema

const = sp.solve(sp.Eq(y_n.rhs.subs(t,0),1))[0]
y_n = y_n.rhs.subs("C1",const)

#Aplicar o operador P(D) no resultado obtido em y_n
h_t =  (1/(R))*sp.diff(y_n,t)
h_t = round_expr(h_t,3)
display(h_t)
h_t = sp.simplify(sp.combsimp(h_t.doit())) + (1/(R))*sp.DiracDelta(t)
display(Math("h_t(t) ={}"+ sp.latex(h_t) + " , t\geq0"))
```

$$\int\limits_{0}^{t} \left(-15.625e^{12.5\tau - 12.5t} + 1.25\delta\left(-\tau + t\right)\right) x(\tau)\, d\tau \tag{12}$$

Calculando a integral de convolução (12) no Python, podemos visualizar a curva de $I(t)$ X t. Plotando a função resultante, foi obtido o seguinte resultado:

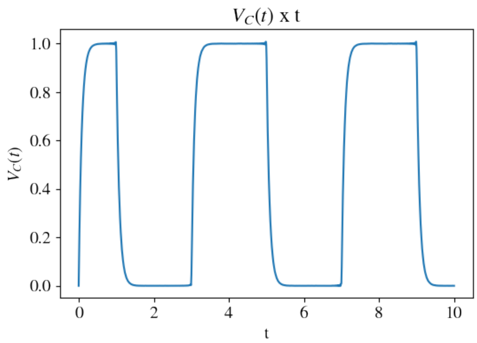


Código Python utilizado para plot do sinal $I(t)$:

```
I_t = sp.Function("I_t")
I_t = resposta_estado_nulo(x,h_t)
corrente = sp.lambdify(t, I_t,
modules = [{'Heaviside': lambda x: np.heaviside(x,1)}, 'numpy'] )
x_vals = np.linspace(0, 10, 1000)
y_vals = (corrente(x_vals))
mpl.plot(x_vals, y_vals)
mpl.ylabel("I(t)")
mpl.xlabel("t")
```

```
mpl.title("I(t) x t")
mpl.show()
```

Conforme apresentado na Seção 2, o gráfico de $V_C(t)$ foi obtido a partir da convolução da resposta ao impulso unitário $h(t)$ com o sinal de entrada $V(t)$. Importante notar que o sinal de entrada V(t) foi representado pela série de Fourier trigonométrica composta por 100 termos (N=100). Podemos observar o plot abaixo:



Código Python utilizado para plot do sinal $V_C(t)$:

```
tensao = sp.lambdify(t, V_c,
modules = [{'Heaviside': lambda x: np.heaviside(x,0)}, 'numpy'] )
x_vals = np.linspace(0, 10, 1000)
y_vals = (abs(tensao(x_vals)))
mpl.plot(x_vals, y_vals)
mpl.ylabel("$V_C(t)$")
mpl.title("$V_C(t)$ x t")
mpl.xlabel("t")
mpl.show()
```

4 Resolução do problema 3)

Foi possível obter os espectros de V(t), partindo dos Harmônicos da série de Fourier previamente calculados, através das Equações descritas em (13).

$$\begin{aligned} c_k &= |X_k| \\ \theta_k &= -\tan^{-1}(b_k/a_k) \end{aligned} \tag{13}$$

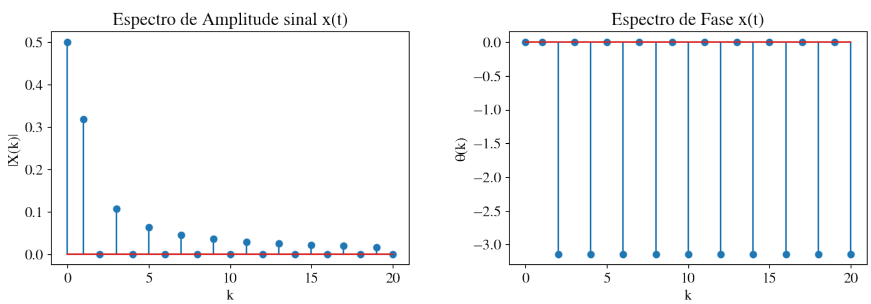


Figura 1: Espectros do Sinal $V(t)$

Através do código Python a seguir, foi possível obter o espectro de amplitude para o sinal V(t), como descrito em (13)

```
#Espectro de Amplitude
x_vals = np.linspace(0, 20, 21)
y_vals = []
```

```
for i in x_vals:
  y_vals.append(abs(Xks.subs(k,i)))

mpl.stem(x_vals, y_vals)
mpl.ylabel("|X(k)|")
mpl.xlabel("k")
mpl.title("Espectro de Amplitude sinal x(t)")
mpl.show()
```

Através do código Python a seguir, foi possível obter o espectro de fase para o sinal V(t),como descrito em (13)

```
#Espectro de Fase
x_vals = np.linspace(0, 20,21)
y_vals = []

for i in x_vals:
  y_vals.append( sp.atan(( 2*sp.im( Xks.subs(k,i)) )/(2*sp.re(Xks.subs(k,i)))) )
  if y_vals[len(y_vals)-1] == sp.nan:
    y_vals[len(y_vals)-1] = -np.pi

mpl.stem(x_vals, y_vals,use_line_collection=True)
mpl.ylabel("theta(k)")
mpl.xlabel("k")
mpl.title("Espectro de Fase x(t)")
mpl.show()
```

Foi possível obter os espectros de I(t), partindo dos Harmônicos encontrados pela resposta em frequência da corrente (14) .

$$H(j\omega) = \int_{-\infty}^{\infty} h(\tau)e^{-j\omega\tau}d\tau \implies H\left(jk\omega_0\right) = \frac{jk\omega_0/R}{jk\omega_0 + 1/RC} \tag{14}$$

$$X_{ktil} = \sum_{k\in\mathbb{Z}} X_k H\left(jk\omega_0\right) \tag{15}$$

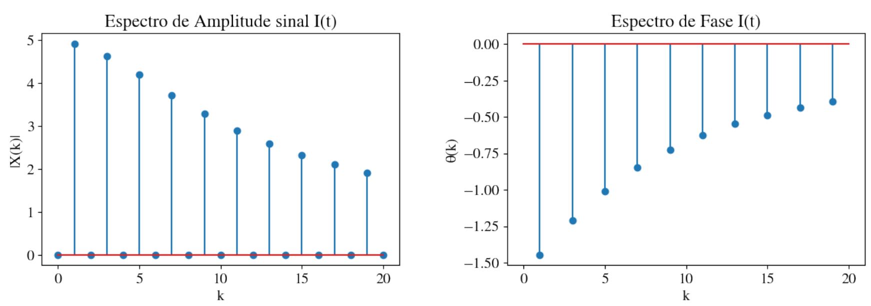


Figura 2: Espectros do Sinal $I(t)$

Através do código Python a seguir, foi possível obter os harmônicos partindo da reposta em frequência para I(t), como descrito em (14) e (15)

```
I_t = 0
X_k_til= 0
for i in range(1,100):
  I_t += (Xks.subs(k,i)*((sp.I*k*w0/R)/ (sp.I*k*w0+ (1/(R*C))))*sp.exp(sp.I*k*w0*t)).subs(k,i)
  X_k_til += (Xks*((sp.I*k*w0/R)/ (sp.I*k*w0+ (1/(R*C))) ))
I_t = round_expr(I_t,3)
```

Através do código Python a seguir, foi possível obter o espectro de amplitude para o sinal I(t)

```
#Espectro de Amplitude
x_vals = np.linspace(0, 20, 21)
y_vals = []

for i in x_vals:
  y_vals.append(abs(X_k_til.subs(k,i)))

mpl.stem(x_vals, y_vals)
mpl.ylabel("|X(k)|")
mpl.xlabel("k")
mpl.title("Espectro de Amplitude sinal I(t)")
mpl.show()
```

Através do código Python a seguir, foi possível obter o espectro de fase para o sinal I(t)

```
#Espectro de Fase
x_vals = np.linspace(0, 20,21)
y_vals = []

for i in x_vals:
  y_vals.append( -sp.atan( (2*sp.im( X_k_til.subs(k,i))) / (2*sp.re(X_k_til.subs(k,i)))))

mpl.stem(x_vals, y_vals,use_line_collection=True)
mpl.ylabel("theta(k)")
mpl.xlabel("k")
mpl.title("Espectro de Fase I(t)")
mpl.show()
```

5 Resolução do problema 4)

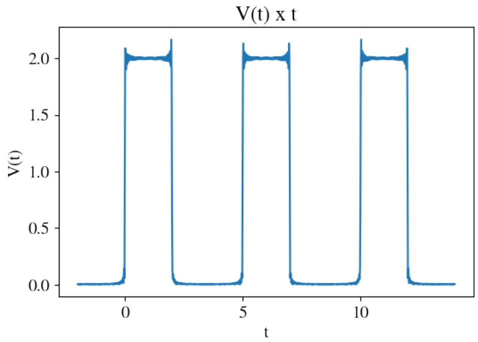
Analogamente, ao procedimento adotado durante a Seção 2 para obtenção da série de Fourier Trigonométrica do sinal de entrada x(t), foram obtidos os harmônicos da série X_k, através da integral (16)

$$0.2\int_0^2 2e^{-\frac{2i\pi kt}{5}}\,dt \implies \begin{cases} \frac{1.0i\left(-1+e^{-\frac{4i\pi k}{5}}\right)}{\pi k} & \text{for } k > -\infty \wedge k < \infty \wedge k \neq 0 \\ 0.8 & \text{otherwise} \end{cases} \tag{16}$$

Através do código Python a seguir, foi possível definir e calcular os harmônicos da série de Fourier de x(t), como exposto em (16)

```
t , j , k , u, w = symbols('t j k u w')
T0 = 5
Ts = 2
w0 = 2*sp.pi/T0
x = 2
auto = sp.exp(-sp.I*k*w0*t)
Xk = (1/T0)*sp.Integral(x*auto,(t,0,Ts))
Xks = sp.simplify(sp.combsimp(Xk.doit()))
Xks
```

Com os harmônicos da série já calculados, foi possível obter a representação da série de Fourier na forma trigonométrica para o novo sinal de entrada x(t) e finalmente, plotar a função resultante tomando N=100 termos na composição da série:



Através do código Python a seguir, foi possível calcular a representação de série de Fourier trigonométrica para o novo sinal de entrada x(t)

```
# Série de Fourier Trigonométrica
x = sp.Function("x")
x = 0
#X0
x =  Xks.subs(k,0)
#Somatória
for i in range(1,100):
  x += ( 2*sp.re(Xks.subs(k,i))*sp.cos(k*w0*t)
  \- 2*sp.im(Xks.subs(k,i))*sp.sin(k*w0*t)).subs(k,i)
x = round_expr(x,3)
lam_x = sp.lambdify(t, x, modules=['numpy'])
x_vals = np.linspace(-1, 2*T0, 1000)
y_vals = (abs(lam_x(x_vals)))
mpl.plot(x_vals, y_vals)
mpl.ylabel("x(t)")
mpl.xlabel("t")
mpl.title("x(t) x t")
mpl.show()
```

Analogamente ao procedimento descrito na Seção 3, foi possível obter, partindo da série de Fourier do sinal de entrada, os gráficos dos espectros de x(t)

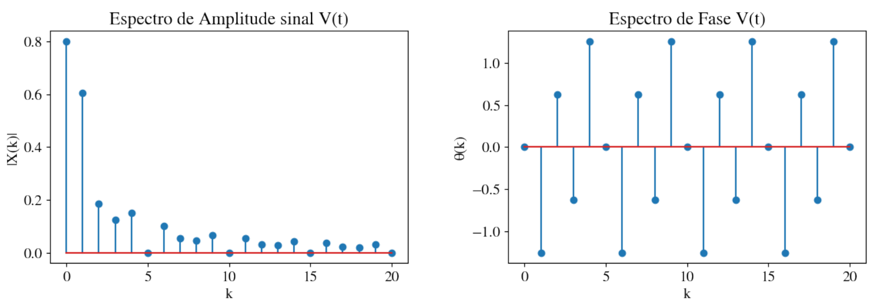
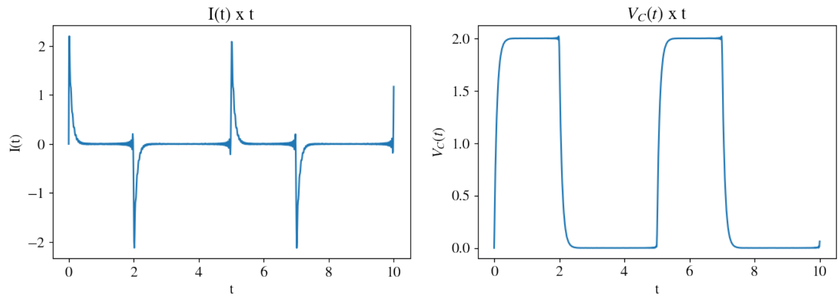


Figura 3: Espectros do Sinal $V(t)$

Para os sinais de $I(t)$ e $V_c(t)$, foram obtidos os seguintes gráficos:



Analogamente ao procedimento da Seção 3, obtemos os espectros do sinal $I(t)$, contidos nos seguintes gráficos:

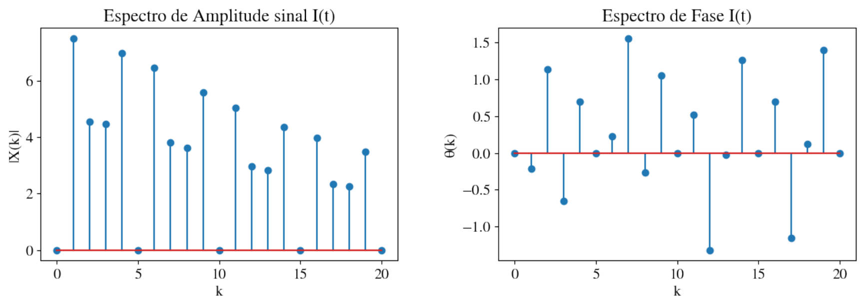
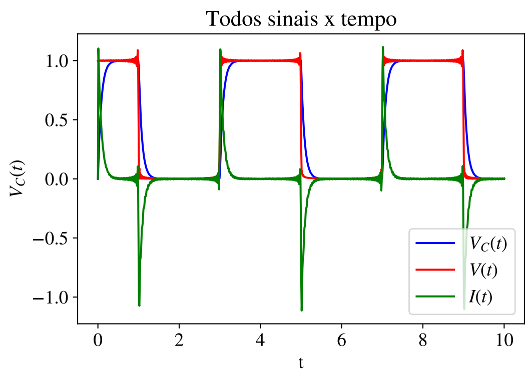


Figura 4: Espectros do Sinal $I(t)$

6 Discussão dos Resultados

Analisando os resultados obtidos nas seções 2 e 3, podemos notar que o número de termos que compõem a série de Fourier (N) é um parâmetro bastante relevante para convergência da série em relação ao sinal original, conforme exposto pelo comparativo 1.

Notamos ainda, que os gráficos de $I(t)$ e $V_c(t)$, obtidos através da série de Fourier são sinais periódicos que acompanham a frequência do sinal original x(t), conforme podemos observar no Comparativo (2), a seguir:



Comparativo 2: Sinais Obtidos nas Seções 2, 3 e 4 em função do tempo

Fisicamente, observamos que quando a tensão fornecida pela fonte de entrada decai, a corrente do circuito passa a se tornar negativa, mostrando que o capacitor passa a fornecer corrente para o circuito. A partir deste momento, a tensão sobre o capacitor V_c passa a decair, com um atraso com relação a tensão da fonte de entrada x(t), até que atinja o valor nulo.

Este comportamento se repete periodicamente, ao longo do tempo, de acordo com a tensão de entrada. Quando a tensão de entrada sobe, o capacitor passa a se recarregar, operando num ciclo contínuo de carga e descarga, como ilustrado pelo Comparativo (2).

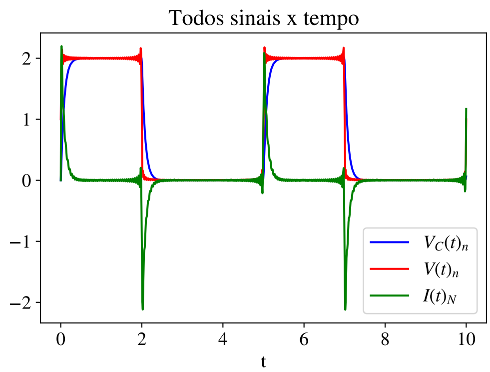
Durante a seção 4, podemos observar na Figura 1 que o espectro de amplitude de x(t) decai de 0.5 até menos de 0.1 no intervalo de k variando entre 0 e 20. Notamos ainda que os k harmônicos pares, não possuem relevância para amplitude da série, assumindo valor 0.

Igualmente, podemos observar na Figura 2 que o espectro de amplitude para $I(t)$ também decai com o aumento do número de harmônicos na série (k), porém com menor velocidade relativamente ao espectro de magnitude de x(t), novamente, observamos que os harmônicos pares não possuem relevância para a amplitude do sinal $I(t)$.

Analisando o resultado obtido na Seção 5, observamos nas Figuras 3 e 4 que os harmônicos múltiplos de 5, não possuem relevância para série de Fourier, tanto no espectro de amplitude quanto no espectro de fase dos sinais $V(t)$ e $I(t)$.

Na figura 3, podemos observar que o espectro de amplitude de $V(t)$ decai de 0.8 até menos de 0.1 no intervalo de k variando entre 0 e 20. Notamos ainda,a partir do espectro de amplitude da Figura 4, que a amplitude dos harmônicos decai periodicamente, a cada 5 harmônicos calculados.

Analogamente aos sinais anteriores, temos que os gráficos de $I(t)$ e $V_c(t)$, obtidos através da série de Fourier, são sinais periódicos que acompanham a frequência do sinal de entrada $V(t)$, conforme podemos observar no comparativo (3), a seguir:



Comparativo 3: Sinais Obtidos na Seção 5 em função do tempo

Por fim, podemos notar que devido à presença de parte imaginária nos harmônicos da série de Fourier obtidos na Seção 5, para o sinal de entrada, - representados pela Equação (16) - temos que os espectros de fase e amplitude se tornam menos regulares relativamente aos espectros obtidos na Seção 4, resultantes de harmônicos puramente reais, isto é, sem parte imaginária.

7 Código na Íntegra

1. **Código Python Utilizado - Repositório GitHub**