**EXEMPLO 1 (Capítulo 4)**

import sympy as sp

def conv(f\_u, g\_x\_minus\_u, a, b, c, d):

# Definindo as variáveis simbólicas

x, u = sp.symbols('x u')

# Função auxiliar para calcular a integral e armazenar tanto os limites quanto os resultados

def calcular\_integral\_com\_limites(f\_u, g\_x\_minus\_u, u\_inf\_str, u\_sup\_str):

# Usando eval para avaliar os limites de integração dinamicamente

u\_inf = eval(u\_inf\_str)

u\_sup = eval(u\_sup\_str)

# Calculando a integral simbólica

integral = sp.integrate(f\_u \* g\_x\_minus\_u, (u, u\_inf, u\_sup))

integral\_simplificado = sp.simplify(integral)

return u\_inf, u\_sup, integral\_simplificado

# Definindo os intervalos de sobreposição baseados em a, b, c, d

intervalos = []

# Intervalo 1: x em [a + c, b + c]

if a + c < b + c:

intervalos.append((a + c, b + c, '0', 'x - c')) # Limites para o intervalo [a + c, b + c]

# Verificação se a + d não é igual a b + c, o que cria mais um intervalo

if a + d != b + c:

# Se a + d for maior que b + c

if a + d > b + c:

intervalos.append((b + c, a + d, 'x - d', 'x - c')) # Intervalo [b + c, a + d]

else:

intervalos.append((a + d, b + c, 'x - d', 'x - c')) # Intervalo [a + d, b + c]

# Intervalo 2: x em [b + c, b + d]

if b + c < b + d:

intervalos.append((b + c, b + d, 'x - d', 'b')) # Limites para o intervalo [b + c, b + d]

# Arrays para armazenar os resultados

x\_vals = []

u\_inf\_vals = []

u\_sup\_vals = []

integrais = []

# Calculando as integrais e armazenando os resultados

for x\_inf, x\_sup, u\_inf\_str, u\_sup\_str in intervalos:

# Calculando a integral para cada intervalo de sobreposição

u\_inf, u\_sup, integral\_simplificado = calcular\_integral\_com\_limites(f\_u, g\_x\_minus\_u, u\_inf\_str, u\_sup\_str)

# Armazenando os dados para resultados

x\_vals.append((x\_inf + x\_sup) / 2) # ponto médio de cada intervalo

u\_inf\_vals.append(u\_inf)

u\_sup\_vals.append(u\_sup)

integrais.append(integral\_simplificado)

# Exibindo as integrais

print(f"\nIntervalo x em [{x\_inf}, {x\_sup}]:")

print(f"Limites de integração: t\_inf = {u\_inf}, t\_sup = {u\_sup}")

print(integral\_simplificado)

# Exemplo de uso da função conv

x, u = sp.symbols('x u')

f\_u = 1 # Definimos f(u) como constante 1

g\_x\_minus\_u = 1 # Definimos g(x - u) como constante 1

a, b = 0, 1 # Intervalo de suporte para f(t)

c, d = 0, 1 # Intervalo de suporte para g(x - t)

conv(f\_u, g\_x\_minus\_u, a, b, c, d)

**EXEMPLO 2 (Capítulo 4)**

import sympy as sp

def conv(f\_u, g\_x\_minus\_u, a, b, c, d):

# Definindo as variáveis simbólicas

x, u = sp.symbols('x u')

# Função auxiliar para calcular a integral e armazenar tanto os limites quanto os resultados

def calcular\_integral\_com\_limites(f\_u, g\_x\_minus\_u, u\_inf, u\_sup):

# Calculando a integral simbólica

integral = sp.integrate(f\_u \* g\_x\_minus\_u, (u, u\_inf, u\_sup))

integral\_simplificado = sp.simplify(integral)

return u\_inf, u\_sup, integral\_simplificado

# Definindo os intervalos de sobreposição baseados em a, b, c, d

intervalos = []

# Caso 1: Intervalo de -1 até x

if a + c < b + c:

intervalos.append((a + c, b + c, a, x)) # Limites para o intervalo [-1, x]

# Verificação para intervalo [b + c, b + d], levando em conta os casos com infinito

if (a + d != b + c) and (b != sp.oo) and (d != sp.oo):

# Se a + d for maior que b + c

if a + d > b + c:

intervalos.append((b + c, a + d, a, x)) # Intervalo [b + c, a + d]

else:

intervalos.append((a + d, b + c, a, x)) # Intervalo [a + d, b + c]

# Caso 2: Intervalo [x, b + d]

if b + c < b + d:

intervalos.append((b + c, b + d, a, b)) # Limites para o intervalo [-1, 1]

# Lista para armazenar os resultados das integrais

partes\_da\_funcao = []

# Calculando as integrais e armazenando os resultados

for x\_inf, x\_sup, u\_inf, u\_sup in intervalos:

# Calculando a integral para cada intervalo de sobreposição

u\_inf, u\_sup, integral\_simplificado = calcular\_integral\_com\_limites(f\_u, g\_x\_minus\_u, u\_inf, u\_sup)

# Gerando a string com a integral e os intervalos

partes\_da\_funcao.append(f"{integral\_simplificado}, {x\_inf} < x < {x\_sup}")

# Exibindo o resultado

return partes\_da\_funcao

# Exemplo de uso da função conv

x, u = sp.symbols('x u')

f\_u = 1 # Função f(u) como constante 1

g\_x\_minus\_u = sp.exp(-(x - u)) # Função g(x - u) com o sinal negativo

a, b = -1, 1 # Intervalo de suporte para f(u)

c, d = 0 , sp.oo # Intervalo de suporte para g(x - u) com infinito (sp.oo)

# Calculando a convolução e exibindo diretamente no Jupyter

resultado = conv(f\_u, g\_x\_minus\_u, a, b, c, d)

# Exibindo o resultado

for parte in resultado:

print(parte)

**EXEMPLO 3 (Avaliação da Unidade II – 2024.2)**

import sympy as sp

def conv(f\_u, g\_x\_minus\_u, a, b, c, d):

# Definindo as variáveis simbólicas

x, u = sp.symbols('x u')

# Função auxiliar para calcular a integral e armazenar tanto os limites quanto os resultados

def calcular\_integral\_com\_limites(f\_u, g\_x\_minus\_u, u\_inf\_str, u\_sup\_str):

# Usando eval para avaliar os limites de integração dinamicamente

u\_inf = eval(u\_inf\_str)

u\_sup = eval(u\_sup\_str)

# Calculando a integral simbólica

integral = sp.integrate(f\_u \* g\_x\_minus\_u, (u, u\_inf, u\_sup))

integral\_simplificado = sp.simplify(integral)

return u\_inf, u\_sup, integral\_simplificado

# Definindo os intervalos de sobreposição baseados em a, b, c, d

intervalos = []

# Intervalo 1: x em [a + c, b + c]

if a + c < b + c:

intervalos.append((a + c, b + c, 'a', 'x - c')) # Limites para o intervalo [a + c, b + c]

# Verificação para intervalo [b + c, b + d], levando em conta os casos com infinito

if (a + d != b + c) and (b != sp.oo) and (d != sp.oo):

# Se a + d for maior que b + c

if a + d > b + c:

intervalos.append((b + c, a + d, a, x)) # Intervalo [b + c, a + d]

else:

intervalos.append((a + d, b + c, a, x)) # Intervalo [a + d, b + c]

# Intervalo 2: x em [b + c, b + d]

if b + c < b + d:

intervalos.append((b + c, b + d, 'x - d', 'b')) # Limites para o intervalo [b + c, b + d]

# Lista para armazenar os resultados das integrais

partes\_da\_funcao = []

# Calculando as integrais e armazenando os resultados

for x\_inf, x\_sup, u\_inf\_str, u\_sup\_str in intervalos:

# Calculando a integral para cada intervalo de sobreposição

u\_inf, u\_sup, integral\_simplificado = calcular\_integral\_com\_limites(f\_u, g\_x\_minus\_u, u\_inf\_str, u\_sup\_str)

# Gerando a string com a integral e os intervalos

partes\_da\_funcao.append(f"{integral\_simplificado}, {x\_inf} < x < {x\_sup}")

# Exibindo o resultado

return partes\_da\_funcao

# Exemplo de uso da função conv

x, u = sp.symbols('x u')

f\_u = 1-u # Função f(t) como t \* exp(-u)

g\_x\_minus\_u = 1 # Função g(x - u) com o sinal negativo

a, b = 0, 1 # Intervalo de suporte para f(u)

c, d = 0, 1 # Intervalo de suporte para g(x - u)

# Calculando a convolução e exibindo diretamente no Jupyter

resultado = conv(f\_u, g\_x\_minus\_u, a, b, c, d)

# Exibindo o resultado

for parte in resultado:

print(parte)