



Sinais e Sistemas - SEL0383/SEL0604

2º semestre/2024

Prof. Marcos Rogério Fernandes

Operação de Convolução em Python

Pedro Gasparelo Leme- 14602421

Vitor Alexandre Garcia Vaz- 14611432

São Carlos - São Paulo

12/12/2024

Introdução

Dentro do contexto de tratamento e processamento de sinais, tem-se a utilização frequente da operação de convolução. A convolução utiliza de duas funções, onde no contexto de sinais tem-se o sinal de entrada $f(t)$ e o filtro aplicado ao sistema $g(t)$, ambos em função do tempo.

Matematicamente, a convolução contínua entre as funções $f(t)$ e $g(t)$ é definida como:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

A importância da convolução está em sua capacidade de descrever como um sistema linear reage a um sinal de entrada, destacando certas características dos sinais. Podendo, dessa forma aplicar filtros passa-alta e passa baixa, fazer modelagem de sistemas além de facilitar a prever como o sistema reagirá a um dado sinal.

Portanto, a convolução facilita a análise de sistemas e sinais complexos, auxiliando em diversas áreas da ciência e engenharia modernas.

Para explorar sua aplicação prática, foi desenvolvido um código em Python que implementa a convolução de sinais de forma simples e eficaz, permitindo compreender e visualizar como essa operação pode ser aplicada no contexto de sinais, facilitando o seu uso e cálculo.

Objetivos

Dessa forma será explorado a operação de convolução no contexto de sinais e sistemas por meio do desenvolvimento de um código em python que efetua a operação mencionada para uma função dada pelo usuário, fornecendo, além disso, o gráfico correspondente às funções fornecidas e ao resultado da convolução efetuado.

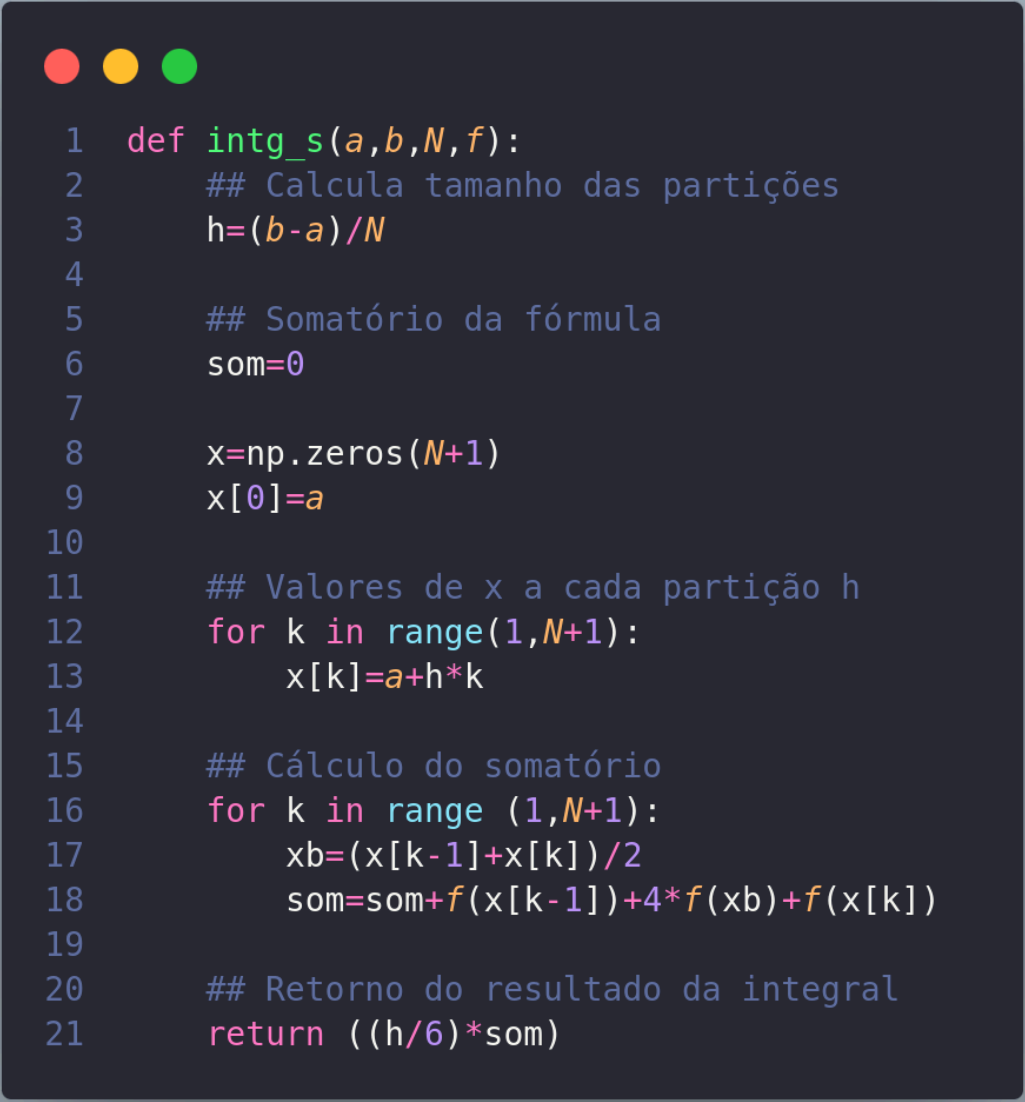
Solução Computacional

Primeiramente, temos a definição da operação de integral, a qual será tratada por meio do método de Simpson, onde, o método fornece o resultado direto das integrais definidas. O método é amplamente utilizado por conta de sua precisão superior quando comparado com outros métodos mais simples, tais como o retângulo ou trapézio. A sua equação é dada por:

$$\int_a^b f(x)dx \approx \frac{h}{3} [f(a) + 4 \sum_{\text{ímpares}} f(x) + 2 \sum_{\text{pares}} f(x) + f(b)]$$

Sendo h a repartição do intervalo.

O código de tal operação é descrito como:



```

1  def intg_s(a,b,N,f):
2      ## Calcula tamanho das partições
3      h=(b-a)/N
4
5      ## Somatório da fórmula
6      som=0
7
8      x=np.zeros(N+1)
9      x[0]=a
10
11     ## Valores de x a cada partição h
12     for k in range(1,N+1):
13         x[k]=a+h*k
14
15     ## Cálculo do somatório
16     for k in range(1,N+1):
17         xb=(x[k-1]+x[k])/2
18         som=som+f(x[k-1])+4*f(xb)+f(x[k])
19
20     ## Retorno do resultado da integral
21     return ((h/6)*som)

```

Figura 1.1- código de descrição da integração pelo método de simpson

Dessa forma, o método de simpson descrito pela Figura 1.1, foi modificado de forma a se adaptar ao procedimento de convolução, o qual é atingido ao realizar o método de simpson com a multiplicação de duas integrais e com a derivada dada pelo $d\tau$, dada por:

```

1  def conv(a,b,N,f,g,x):
2
3      ## Calcula tamanho das partições
4      h=(b-a)/N
5
6      ## Somatório da fórmula
7      som=0
8
9      t=np.zeros(N+1)
10     t[0]=a
11
12     ## Valores de t a cada partição h
13     for k in range(1,N+1):
14         t[k]=a+h*k
15
16     ## Cálculo do somatório
17     for k in range(1,N+1):
18         tb=(t[k-1]+t[k])/2
19         som += f(t[k-1])*g(x-t[k-1]) + 4*f(tb)*g(x-tb) + f(t[k])*g(x-t[k])
20
21     ## Retorno do resultado da convolução no ponto x
22     return ((h/6)*som)

```

Figura 1.2- código de descrição da operação de convolução

Para a utilização da função de convolução, basta o operador definir as funções $f(x)$ e $g(x)$, o número de repartições N , o qual irá dizer a precisão da operação da integral e os limites inferiores e superiores da integral em si, dando preferencialmente um valor elevado, não será utilizado os valores infinitos pois tal operação nunca poderá ser concluída pelo computador. Como por exemplo na Figura 1.3 onde se tem a definição de duas funções Gate 10, além disso tem-se a utilização de `x_values`, o qual é um vetor que irá armazenar os valores de x obtidos pela repartição do intervalo, que serão aplicados na função de simpson.

```

1  # Definindo as funções f e g
2  def f1(t):
3      if(abs(t) <= 5):
4          return 1
5      return 0
6
7  def g1(t):
8      if(abs(t) <= 5):
9          return 1
10     return 0
11
12  # Parâmetros
13  a = -20          # limite inferior
14  b = 20           # limite superior
15  N = 1000         # número de partições
16
17  # Definindo os valores de x
18  x_values = np.linspace(a, b, N)
19
20  # Calculando a convolução para cada valor de x
21  i = 0
22  conv_values = np.zeros(N)
23  for x in x_values:
24      conv_values[i] = conv(a,b,N,f1,g1,x)
25      i += 1

```

Figura 1.3- exemplo de declaração de funções para a convolução pelo usuário

Por fim, tem-se a chamada do procedimento de plotagem do gráfico, o qual irá plotar, em um mesmo plano cartesiano, ambas as funções que serão utilizadas na operação de convolução, e o resultado da convolução em si de ambas as funções, dado pela Figura 1.4, onde os resultados serão explorados na seção, **Resultados**.

```

1  # Plotando f(x)
2  plt.figure(figsize=(10, 6))
3  plt.plot(x_values, [f1(x) for x in x_values], label='f(x) = g(x) = $G_{10}(x)$', color='blue')
4  plt.xlabel('x')
5  plt.ylabel('f(x)')
6  plt.title('Função f(x)')
7  plt.legend()
8  plt.grid(True)
9
10 # Salvamento do gráfico
11 plt.savefig('graficos_plotados/funcao_f1.png')
12
13 # Exibição do gráfico
14 plt.show()
15
16 # Plotando os resultados
17 plt.figure(figsize=(10, 6))
18 plt.plot(x_values, [f1(x) for x in x_values], label='f(x) = $G_{10}(x)$', color='blue')
19 plt.plot(x_values, [g1(x) for x in x_values], label='g(x) = $G_{10}(x)$', color='orange')
20 plt.plot(x_values, conv_values, label='Convolução de f e g', color='#fa19ef')
21 plt.xlabel('x')
22 plt.ylabel('y(x)')
23 plt.title('$y(x) = f(x) * g(x)$')
24 plt.legend()
25 plt.grid(True)
26
27 # Salvamento do gráfico
28 plt.savefig('graficos_plotados/convolucao1.png')
29
30 # Exibição do gráfico
31 plt.show()

```

Figura 1.4- Descrição da operação de plotagem de gráfico após operação de convolução

Resultados

Para a compilação do código, a operação de convolução foi efetuado para 3 casos em específico, sendo eles com as funções sendo:

$$\text{Caso 1: } f(t) = G_{10}(t), \quad g(t) = G_{10}(t)$$

Da resolução da fórmula da convolução obtemos os seguintes valores em função de t:

$$(f * g)(t) = 0 \quad p/ \quad t < -10$$

$$(f * g)(t) = t + 10 \quad p/ \quad -10 < t < 0$$

$$(f * g)(t) = 20 - t \quad p/ \quad 0 < t < 10$$

$$(f * g)(t) = 0 \quad p/ \quad t > 10$$

Dessa forma, pode-se esperar que o resultado seja uma função triangular com base de -10 a 10 e com pico em 0 valendo 10. o gráfico obtido segue a seguir:

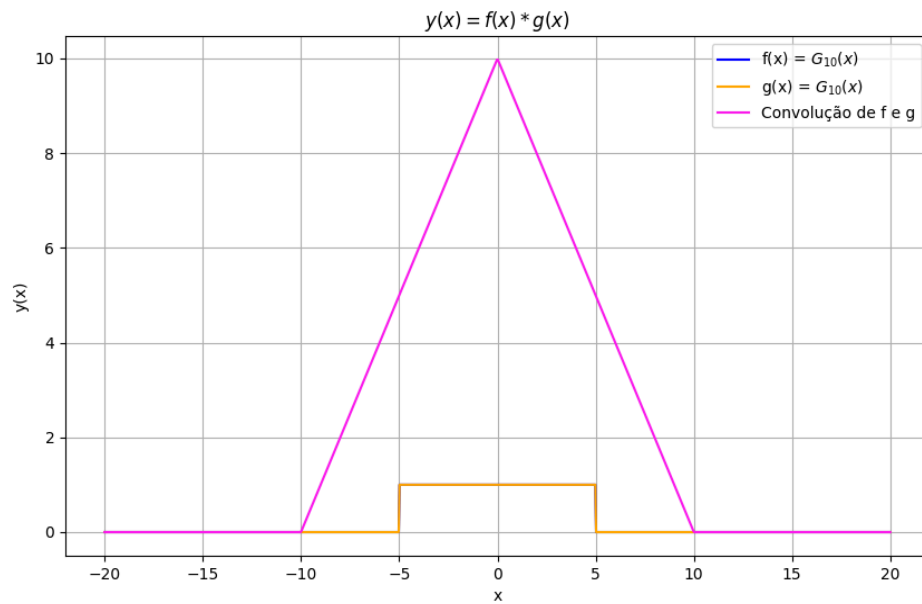


Figura 2.1- convolução de duas funções Gate

Caso 2: $f(t)=\text{Rampa}(t)$, $g(t)= G5(t)$

Da resolução da fórmula da convolução obtemos os seguintes valores em função de t :

$$(f * g)(t) = 0 \quad p/ \quad t < -\frac{5}{2}$$

$$(f * g)(t) = 5t + 12.5 + \frac{(t+2.5)^2}{2} \quad p/ \quad -\frac{5}{2} < t < \frac{5}{2}$$

$$(f * g)(t) = 25 - 5t + \frac{(t-2.5)^2}{2} \quad p/ \quad \frac{5}{2} < t < 5$$

$$(f * g)(t) = 0 \quad p/ \quad t > 5$$

Gráfico obtido da compilação em python:

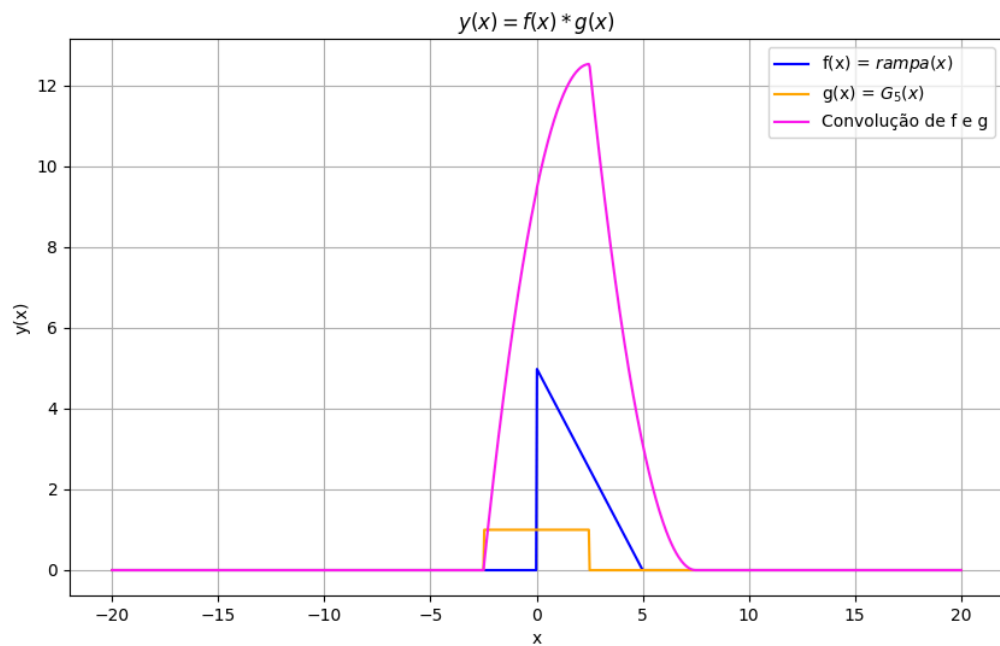


Figura 2.2- convolução de função rampa com função gate 5

Caso 3: $f(t) = e^{0,2t} u(-t)$, $g(t) = e^{-0,2t} u(t)$

Da resolução da fórmula da convolução obtemos os seguintes valores em função de t:

$$(f * g)(t) = \frac{1}{0,4} e^{0,2t} \quad p/t < 0$$

$$(f * g)(t) = \frac{1}{0,4} e^{-0,2t} \quad p/t > 0$$

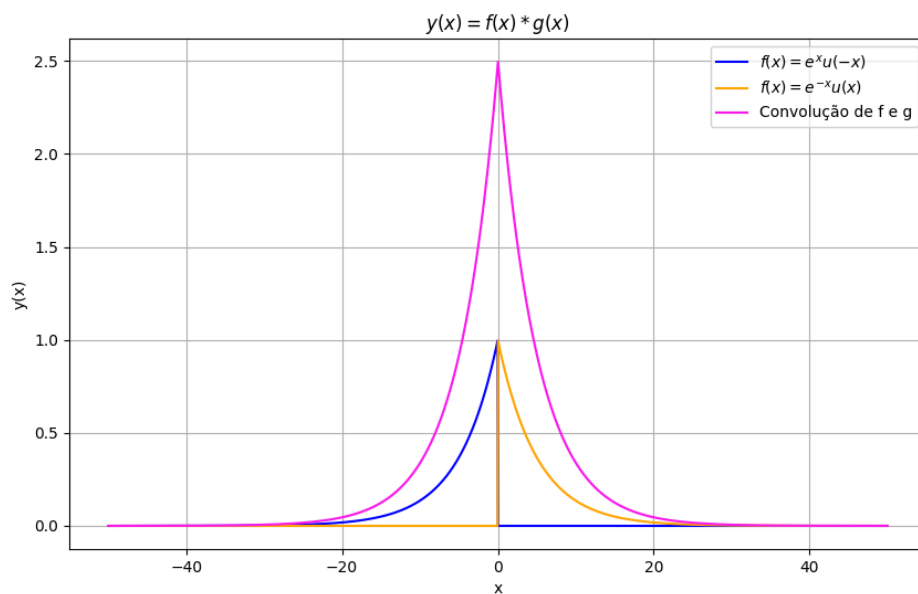


Figura 2.3- convolução entre $f(x) = e^{0,2x} u(x)$ $g(-x) = e^{-0,2x} u(x)$

Para melhor visualização de $f(x)$ e $g(x)$:

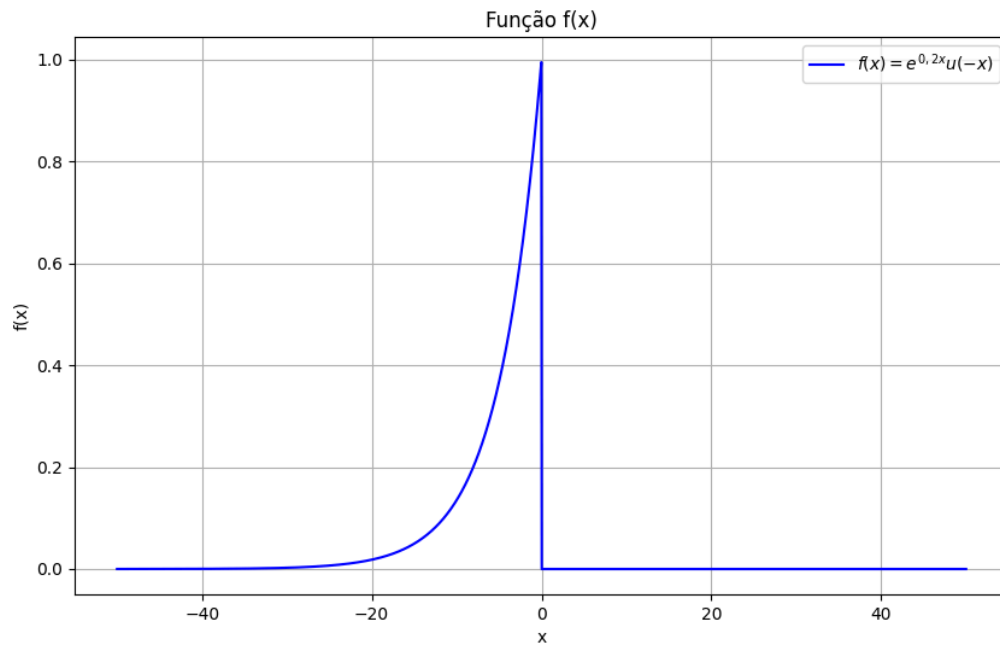


Figura 2.4- Gráfico de $f(x) = e^{0,2x}u(-x)$

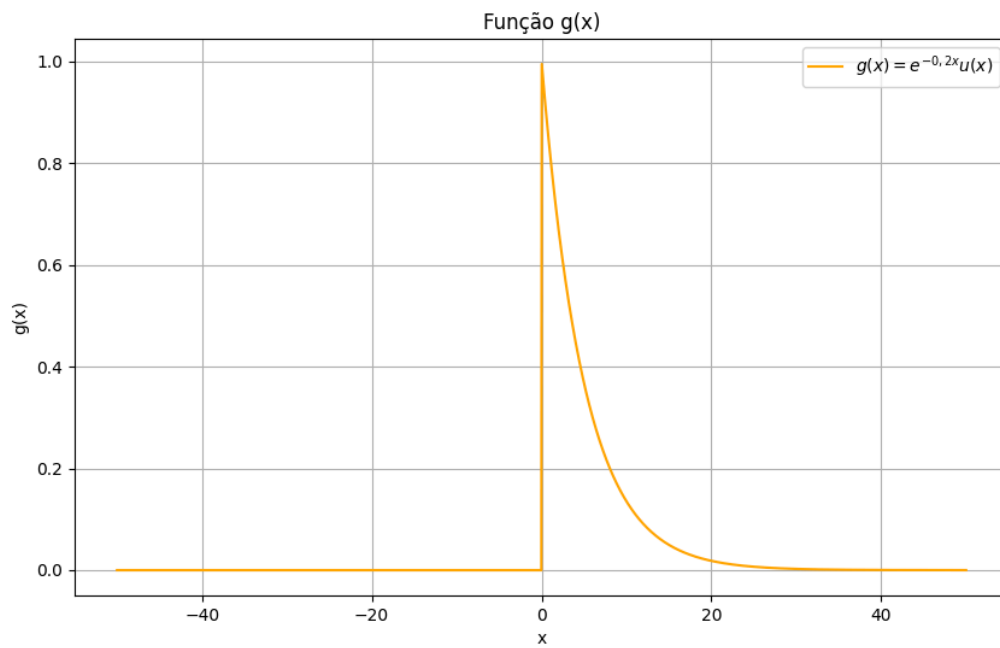


Figura 2.5- Gráfico de $g(x) = e^{-0,2x}u(x)$

Conclusão

Dos Resultados obtidos pela compilação do código e da geração dos gráficos dados pelas Figuras 2.1 a 2.3 e dos resultados das convoluções realizados manualmente para a aferição da veracidade do código, pode-se observar que os resultados obtidos são verídicos, indicando o bom funcionamento do código para a execução da convolução para diferentes funções dadas pelo operador.

Para mais detalhes sobre o trabalho computacional, estamos disponibilizando um repositório no github, o qual foi criado para realização dessa atividade: [Link GitHub](#).

Referências

AUSAS, Roberto F. *Notas de aula SMA0305*. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 26 jun. 2024.