Universidade de Brasília
Departamento de Engenharia Elétrica
Comunicações Digitais (167878)

Tarefa 1

Lucas Henrique Santos Souza 140150838
Welligtton Silva Cavedo 130018660

Professor:
Paulo H. Portela de Carvalho

14 de Setembro de 2020

Problema proposto 1

Q1 Implementar em Python ou Octave uma função que gera, dado um intervalo de tempo de verificação desejado T, um sinal cossenoidal discreto cujos parâmetros são: amplitude A, frequência f0, fase p0 e taxa de sobre-amostragem Ra, definida como a razão entre a taxa de amostragem desejada e a taxa mínima de amostragem de Nyquist.

Resolução

Q1 A primeira questão solicita uma construção funcional que gere um sinal cossenoidal na forma $g(t) = Acos(2p\pi f_0 t + p_0)$ considerando um certo parâmetro definido como a razão entre a taxa de amostragem e a taxa mínima de amostragem em Nyquist, a denominada taxa de sobre-amostragem.

Principiemos considerando um sinal analógico arbitrário *g(t)* de Energia finita e especificado para todo instante de tempo t, conforme a figura 1 abaixo.

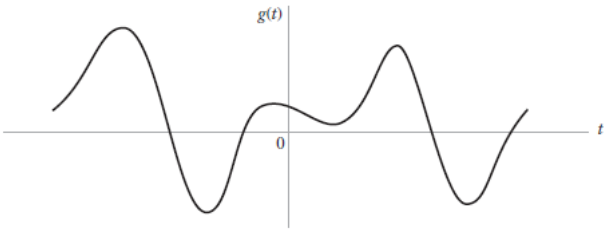


Figura 1: Fonte: Haykin, 2008.

Imaginemos um processo de amostragem instantâneo sobre *g(t)* a uma certa taxa uniforme a cada T_s segundos e, como resultado deste processo, obtenhamos uma sequência infinita de números espaçados por um fator T_s segundos e denotada por $g(n\,T_s)$ onde $n \in \mathbb{Z}$.

Por definição, T_s é o chamado período de amostragem e o seu recíproco, $\frac{1}{T_s}$, é definido como a taxa de amostragem f_s. A figura 2 ilustra o sinal *g(t)* amostrado instantaneamente.

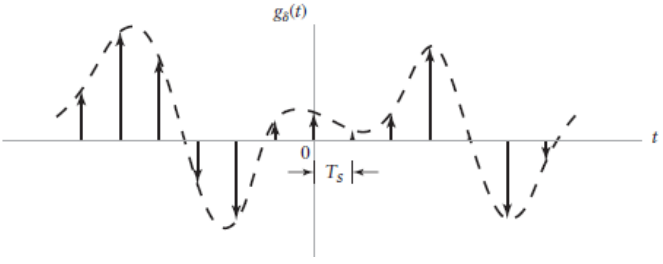


Figura 2: Fonte: Haykin, 2008.

Essa ideia de amostragem já era conhecida dos Matemáticos desde 1915 com Whittaker [1], foi introduzida em Teoria da Comunicação por Shannon [2] em 1949 com o célebre artigo *"Communication in the presence of noise"*, mas de fato o

interesse pelo Teorema da Amostragem por parte dos Engenheiros de Comunicação remonta a Nyquist [3] em 1928, cujas contribuições possibilitaram estamentar o *Teorema da Amostragem* para sinais estritamente limitados em banda e de energia finita em duas partes equivalentes:

- Um sinal limitado em banda e de energia finita que não possui componentes frequenciais maiores do que B hertz é completamente descrito especificando os valores do sinal em instantes de tempo separados por $\frac{1}{2B}$ segundos.
- Um sinal limitado em banda e de energia finita que não possui componentes frequenciais superiores a B hertz é completamente recuperado a partir do conhecimento de suas amostras tomadas a uma taxa de 2B amostras por segundo.

A taxa de amostragem de 2B amostras por segundo para um sinal de largura de banda de B hertz é chamada de taxa de Nyquist; seu recíproco (medido em segundos) é chamado de intervalo de Nyquist. Uma parte analítica que se aplica ao transmissor e outra parte de síntese que se aplica ao receptor.

Finalizando estas considerações iniciais e retomando a ideia do fator de sobreamostragem salienta-se que a taxa de Nyquist é a taxa de amostragem mínima admissível.

Considerando os parâmetros do problema temos destarte:

$$Ra = \frac{f_s}{2f_0} \Rightarrow f_s = 2f_0Ra \tag{1}$$

Testamos para diferentes valores de parametrização. Pelo Teorema, para a reprodução fiel de um sinal contínuo em domínio discreto, deve-se amostrar o sinal a uma taxa f_s maior que pelo menos duas vezes a máxima frequência f_0 contida no sinal.

Começamos então por este limiar mínimo com uma onda de 10Hz e Amplitude unitária, obtendo a figura 3, que não é uma curva suave e fidedigna à representação do cosseno:

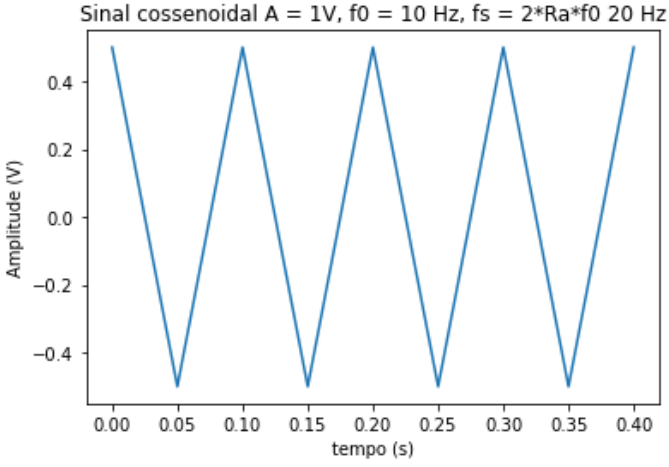


Figura 3: Fonte: Elaborado pelos autores, 2020

Como o Python é uma linguagem interpretada, para obter um processamento digital de uma senoide suave a taxa de amostragem deve ser muito maior que o mínimo prescrito pela relação de Nyquist-Shanon (pelo menos o dobro da máxima frequência f_0 contida no sinal). Portanto precisamos amostrar o sinal de entrada à uma taxa significativamente maior do que aquela que o critério define - aqui entra o fator de sobre-amostragem. Do ponto de vista computacional, no entanto, quanto maior o fator de sobre-amostragem, mais memória é necessária para o armazenamento do sinal.

Um fator de sobre-amostragem de 20 foi escolhido para a função anterior. Dessa forma conseguimos traçar uma onda senoidal contínua e suave, conforme figura 4. Assim, a taxa de amostragem torna-se $f_s = 2f_0Ra = 400$

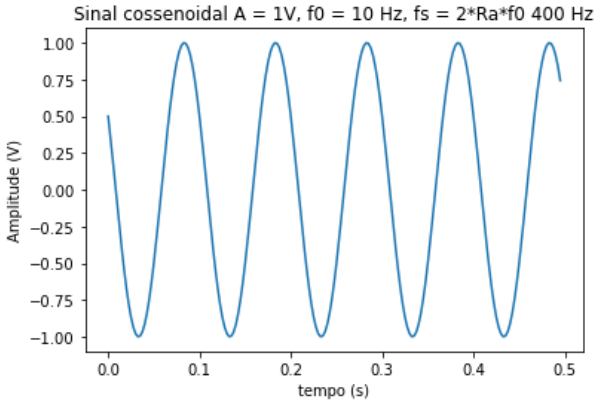


Figura 4: Fonte: Elaborado pelos autores, 2020

Do ponto de vista computacional, no entanto, quanto maior o fator de sobre-amostragem, mais memória é necessária para o armazenamento do sinal.

Problema proposto 2

Q2 Implementar em Python ou Octave uma função que plote um sinal discreto, dado um intervalo de visualização desejado. Aplicar essa função sobre o sinal do item 1, para diferentes valores de parâmetros.

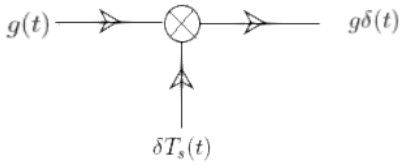
Resolução

Retomando a ideia da forma ideal de amostragem instantânea desenvolvida na resolução do item 1, pensemos agora a sequência de números $g(nT_s)$ como o produto por uma sequência de impulsos unitários espaçados por T_s e cujas contribuições individuais são somadas. Obtemos:

$$\begin{aligned} g\delta(t) &= \sum_{n=-\infty}^{+\infty} g(nT_s)\delta(t - nT_s) \Rightarrow \\ g\delta(t) &= g(t)\delta T_s(t) = g(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s) \end{aligned} \tag{2}$$

Esse modelo matemático de amostrador ideal, ilustrado na figura 5 permite uma forma conveniente para poder ser interpretada por um processador digital. Conforme [4], constitui a ponte entre o mundo a contínuo e o mundo discreto.

Figura 5: Modelo matemático de amostrador ideal



Fonte: Elaborado pelos autores, 2020.

No Python, a própria funcão *stem* do módulo *pyplot* da biblioteca *matplotlib*[1] já oferece o sequenciamento dos valores da série temporal como um conjunto discreto de argumentos, ou seja, plota um gráfico de hastes com linhas verticais em cada localização x da linha de base, colocando um ponto de marcação no eixo y. Dessa forma simplesmente chamamos a função *stem* sobre o sinal gerado no item, obtendo a figura 6. Notar que foram escritas linhas de código adicional para gerar uma saída no Python em alta resolução, mas como o número de amostras é significativo a resolução ainda não ficou aceitável para ser mostrada de forma adequada em uma página sem perda detalhes fundamentais.

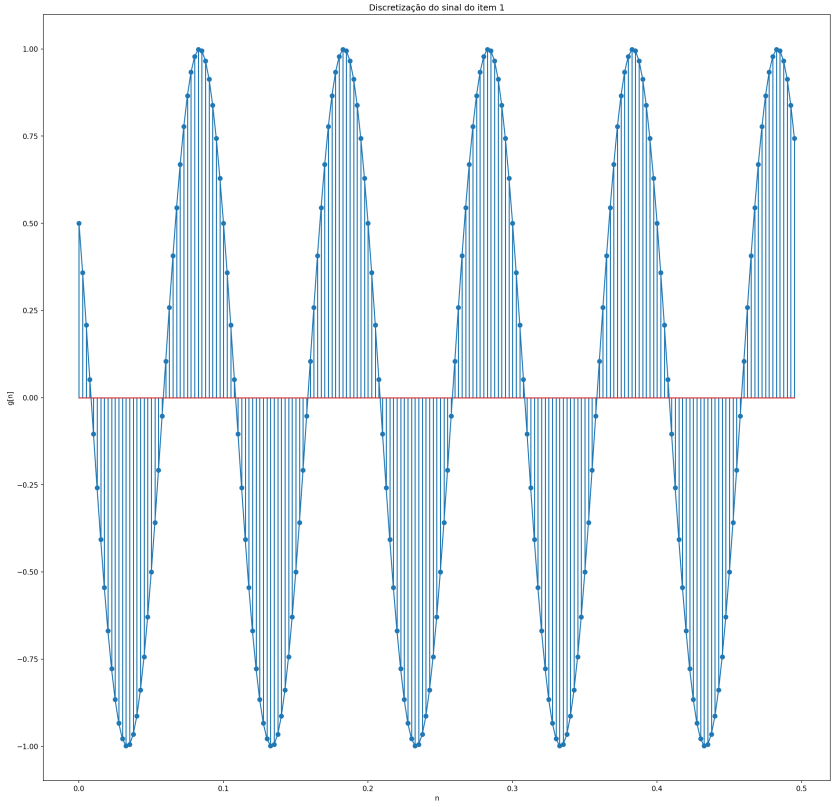


Figura 6: Fonte: Elaborado pelos autores, 2020.

1

[1] https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.stem.html

Problema proposto 3

Q3 Aplicar a Transformada Rápida de Fourier (FFT) sobre a função definida no item 1, para diferentes valores de seus parâmetros, utilizando funções pré programadas em Python e Octave. Plotar e analisar o resultado da FFT e responder: Que relações devo obedecer para que o resultado da FFT seja adequado para uma análise espectral correta de um sinal discreto?

Resolução

A questão solicita a aplicação da FFT. Partindo da conceituação discutida em [5], a Transformada Rápida é um aperfeiçoamento algoritmizado da Transformada Discreta de Fourier (computador digital só trabalha com dados discretos) desenvolvido por Cooley e Tukey em 1965 [6]. Esse algoritmo reduz o número de cálculos da ordem de ${N_0}^2$ para $N_0 log N_0$, tornando a Transformada de Fourier acessível para o processamento digital de sinais.

O Teorema da Amostragem e o seu dual, resumidos na tabela 1, fornecem a relação quantitativa que torna numericamente possível a implementação computacional da Transformada em sua forma discreta (TDF).

Tabela 1: Relação quantitativa na forma da Transformada Discreta

Teorema da Amostragem	Dual
Um sinal limitado em faixa a BHz pode ser reconstruído exatamente de suas amostras se a taxa de amostragem for $fs > 2\text{B}Hz$	Para um sinal limitado em tempo a τ segundos, seu espectro $X(\omega)$ pode ser reconstruído das amostras de $X(\omega)$ tomadas a intervalos uniformes não maiores do que $1/\tau Hz$

Utilizamos a FFT, implementada no pacote *Scipy.fftpack*[2], que faz uso do algoritmo de Cooley-Tucey para o cálculo eficiente da DFT. Começamos explorando a função *fft()* documentada na biblioteca que retorna uma aproximação da DFT com ω (radianos/s) de 0 a π (ou seja, 0 a *fs*, onde *fs* é a frequência de amostragem). Notar que essa função já faz *zero-pad* na entrada com uma grande quantidade de zeros. Utilizamos 1024 pontos amostrais (N pontos da DFT) sobre o mesmo sinal do item 1 de frequência $f_0 = 10$Hz, obtendo a figura 7.

[2] https://docs.scipy.org/doc/scipy/reference/fftpack.html

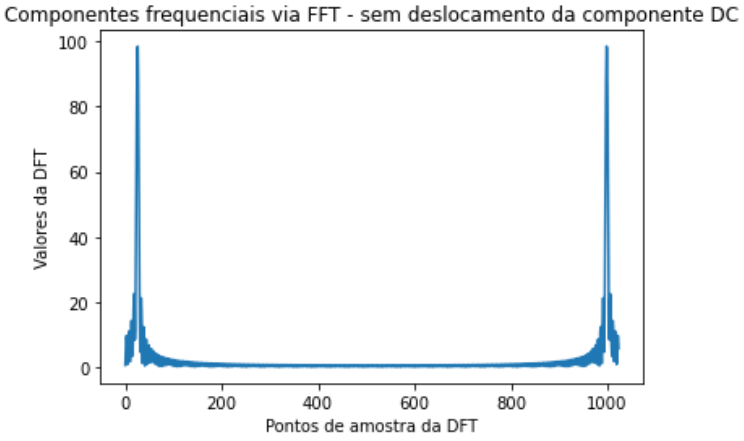


Figura 7: Fonte: Elaborado pelos autores, 2020.

Notar que a análise espectral do sinal não esta correta. Deste gráfico não é possível identificar a frequência f_0 do sinal cossenoidal que geramos no item 1, que, posto ser um sinal real no domínio do tempo tem espectro simétrico conjugado, portanto o conteúdo em 10Hz implica um igual conteúdo em -10Hz. O conteúdo visível no gráfico é apenas imagem *alias* do conteúdo esperado.

Portanto a pergunta é pertinente: "*Que relações devo obedecer para que o resultado da FFT seja adequado para uma análise espectral correta de um sinal discreto?*"

Formalizaremos os seguintes procedimentos:

- No domínio espectral as frequências assumem valores positivos e negativos
- Precisamos então plotar os valores da DFT em um eixo de frequência com valores positivos e negativos e com o valor de indexação 0 posicionado no meio
- A funcão *fftshift* da *scipy* realiza este *shift* de frequência da componente de frequência zero (DC) para o centro do espectro.

Vejamos:

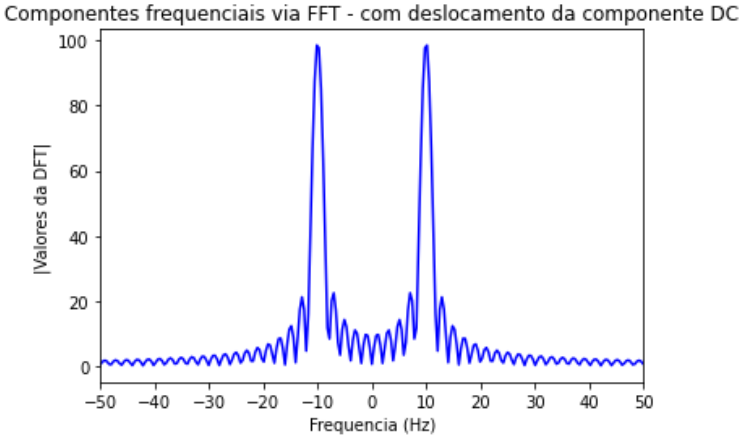


Figura 8: Fonte: Elaborado pelos autores, 2020.

Ao trocar o vetor de saída de *fft()* para o meio, isto é de -$\frac{\pi}{2}$ a $\frac{\pi}{2}$ (via *fftshift()*) o resultado alinha-se com a previsão teórica. Em outras palavras o giro dos resultados de forma que o termo DC esteja no centro, a meio caminho entre -$\frac{fs}{2}$ e $\frac{fs}{2}$, que é a forma mais comum de exibição de espectro, deve ser considerado para uma correta análise espectral.

Problema proposto 4

Q4 Implementar o sinal trem de impulso periódico discreto de período *Ts*. Repetir os itens 2 e 3 para o sinal definido neste item.

Resolução

O trem de impulso periódico é um ferramental extremamente importante para o processamento de sinais e para a teoria da amostragem, que fora brevemente discutida na questão 1.

Na sua forma contínua no tempo é definido como:

$$\delta T_0(t) = \sum_{m=-\infty}^{+\infty} \delta(t - mT_0) \tag{3}$$

Devido à largura infinitamente pequena do impulso, é extremamente complicado modelar numericamente o trem em tempo contínuo. Portanto já iniciamos a discussão partindo para o domínio discreto, onde o tratamento numérico é mais simples. Portanto para um período T_s, tem-se:

$$\delta T_s[n] = \sum_{k=0}^{\frac{L}{Ts-1}} \delta[n - kT_s] \tag{4}$$

Donde, L é divisível por T_s e $\delta T_s[n]$ representa uma função que tem um impulso a cada T_s amostras. Implementamos este processo em Python prontamente, obtendo a figura 9.

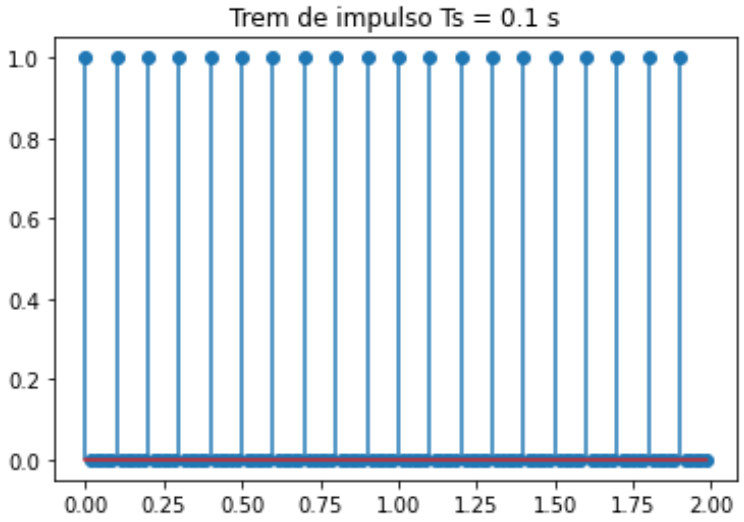


Figura 9: Fonte: Elaborado pelos autores, 2020.

O próximo passo exigido pela questão é a computação da Transformada de Fourier. Voltemos para o domínio contínuo para algumas observações.

A equação 3, que define a chamada função de amostragem ideal, consiste de uma sequência infinita de impulsos uniformemente espaçados, conforme mostrado abaixo:

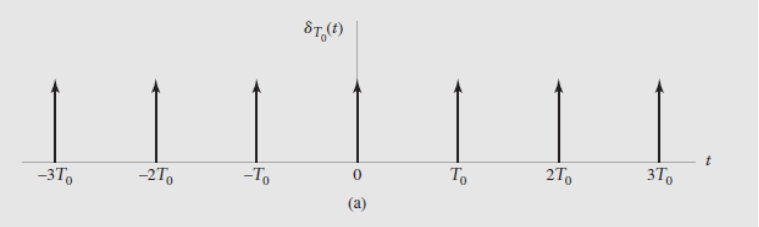


Figura 10: Fonte: Haykin, 2008.

Posto que o trem de impulso periódico simplesmente consiste de impulsos $\delta(t)$ e lembrando do par de Transformadas $1 \rightleftharpoons G(f)$, tem-se que $G(f) = 1$ e $G(\frac{n}{T_0}) = G(nf_0)) = 1 \; \forall n$.

E, mais ainda, a partir do par $e^{j2\pi f_c t} \rightleftharpoons \delta(f - f_c)$

E do par resultante da derivação da Fórmula do Somatório de Poisson engendrado pelo par de Transformada:

$$\sum_{n=-\infty}^{+\infty} g(t - mT_0) \rightleftharpoons f_0 \sum_{n=-\infty}^{+\infty} G(nf_0)\delta(f - nf_0) \tag{5}$$

Chega-se à relação:

$$\sum_{n=-\infty}^{+\infty} g(t - mT_0) \rightleftharpoons f_0 \sum_{n=-\infty}^{+\infty} \delta(f - nf_0) \tag{6}$$

Este resultado mostra que a Transformada de Fourier de um trem de impulso periódico, espaçados a um intervalo de T_0 segundos, consiste em um outro conjunto de impulsos ponderados por um fator $f_0 = \frac{1}{T_0}$ e regularmente espaçados em f_0 Hz ao longo do eixo de frequências. Conforme mostrado abaixo na figura 11:

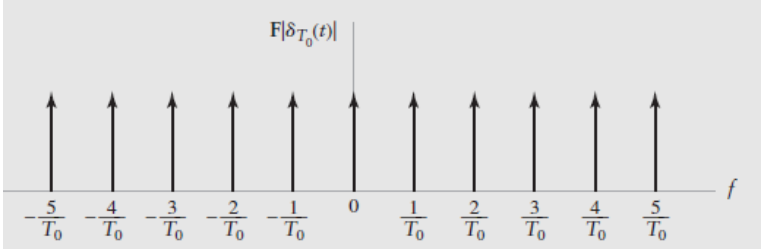


Figura 11: Fonte: Haykin, 2008.

Esta é a previsão teórica esperada também para o nosso caso discreto, salvaguardadas as peculiaridades do domínio.

Aplicamos a FFT em dois ensaios diferentes, variando o parâmetro T_s. Obtivemos o resultado ilustrado na figura 12.

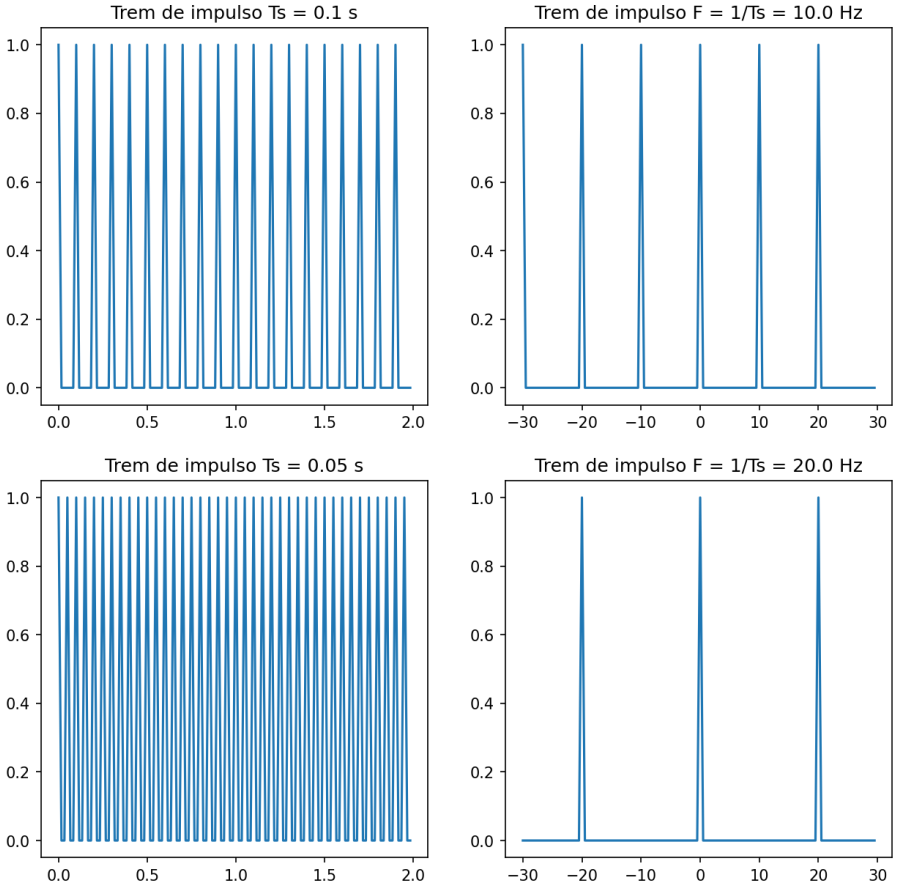


Figura 12: Fonte: Elaborado pelos autores, 2020.

Sumarizando os achados da programação notamos que a multiplicação no domínio do tempo corresponde à uma convolução no domínio da frequência. Multiplicar por um trem de impulso portanto, corresponde a uma convolução com a DFT (ou FFT nesse caso) de um trem de impulso. A convolução com um trem de impulso faz várias cópias do espectro do sinal.

Como se constatou acima, a Transformada do trem também é um trem de impulsos e quanto mais próximos no domínio do tempo, mais espaçados eles se tornam no domínio da frequência. Ou seja, em geral, mais impulsos no domínio do tempo correspondem a menos impulsos no domínio da frequência. Pode-se enxergar a amostragem como uma multiplicação por um trem de impulso.

Problema proposto 5

Q5 Implementar o sinal porta (pulso retangular) discreto de largura Tg e taxa de amostragem Rg. Repetir os itens 2 e 3 para o sinal definido neste item.

Resolução

Uma porta retangular isolada de amplitude A e duração T_g é representada matematicamente no domínio do tempo como:

$$g(t) = Arect\frac{t}{T_g} \tag{7}$$

onde

$$rect(t) = \begin{cases} 1, & -\frac{1}{2} < t < \frac{1}{2} \\ 0, & c.c \end{cases} \tag{8}$$

Para abordar esta questão estudamos e testamos um módulo relativamente recente no Python, a biblioteca *"Signals and Systems Function Module"* - sigsys[3] - documentada na comunidade aberta PyPI e orientada para o processamento digital de sinais.

Para começar, implementamos uma porta retangular de Amplitude unitária e largura T_g customizável, conforme figura 13.

Em seguida, discretizamo-na a uma taxa de amostragem $R_g = 5$, obtendo a saída ilustrada na figura 14, que está um ajuste de resolução para melhorar a visualização.

Finalmente, aplicamos a Transformada rápida - cuja roupagem na sigsys é encapsulada na função *ss.ft_approx*.

Obtivemos a figura 15.

[3] https://pypi.org/project/scikit-dsp-comm/

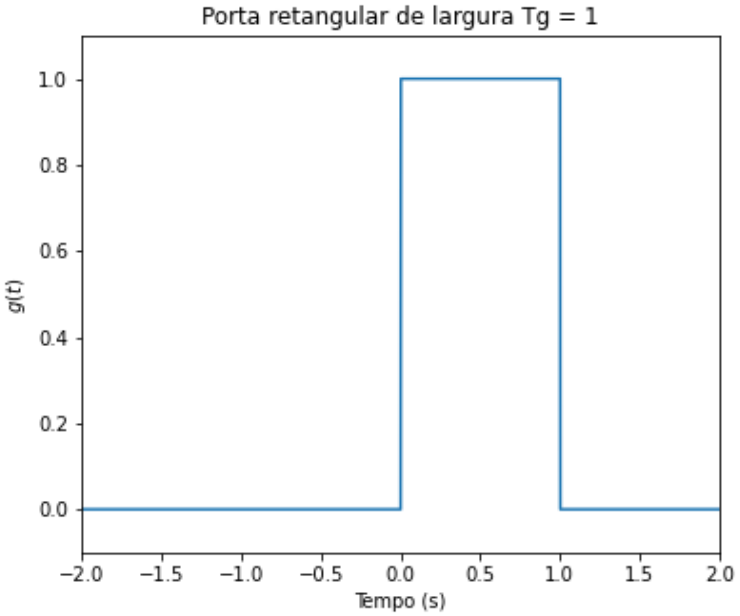


Figura 13: Fonte: Elaborado pelos autores, 2020.

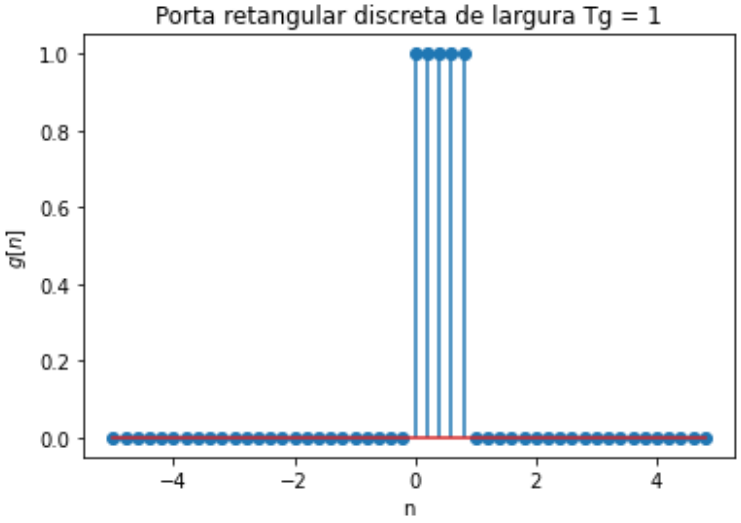


Figura 14: Fonte: Elaborado pelos autores, 2020.

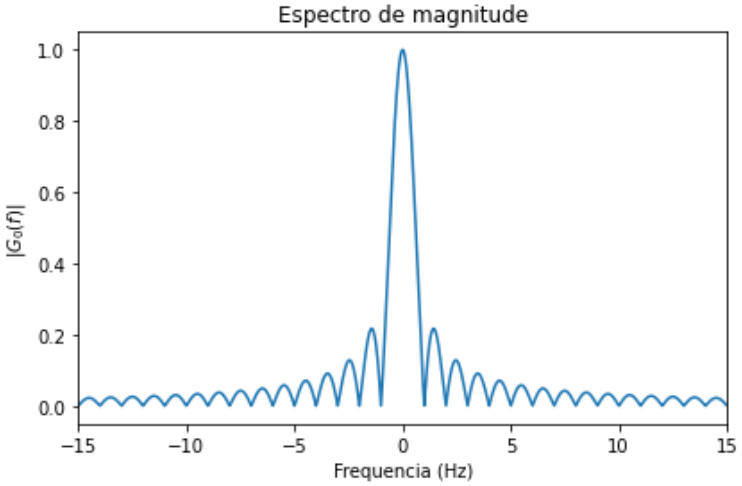


Figura 15: Fonte: Elaborado pelos autores, 2020.

Resultado esperado conforme a previsão teórica, posto conhecermos o par

$$Arect\frac{t}{T_g} \rightleftharpoons AT_g sinc(fT_g) \tag{9}$$

Onde o espectro de amplitude tem um pico em $f = 0$ com valor igual a AT_{g} e os nulos do espectro ocorrendo em múltiplos de $\frac{1}{T_{\mathrm{g}}}$. À medida que a duração do pulso $\mathrm{T_g}$ diminui, este primeiro cruzamento no zero "avança"na frequência e, inversamente, conforme a duração do pulso é aumentada, o primeiro cruzamento de zero se move em direção à origem. Isto mostra a relação inversa entre a descrição no domínio do tempo e a descrição no domínio da frequência, qual seja: um pulso estreito no tempo tem uma descrição significativa em uma ampla gama de frequências no domínio espectral, e vice e versa.

Problema proposto 6

Q6 Definir matematicamente convolução de sinais em tempo discreto. Determinar a convolução do sinal trem de impulso periódico discreto com o sinal porta discreto, para diferentes valores de seus parâmetros constituintes. Neste item, não se está exigindo implementação computacional, mas o bônus de que trata o Plano de Ensino será concedido para o grupo que realizar a implementação computacional da função descrita neste item e realizar a análise do resultado da convolução para diferentes funções e valores de seus parâmetros.

Resolução

Finalmente, a questão 6 solicita uma formalização da convolução para sinais em tempo discreto. A convolução é um operador matemático que toma duas funções x e h e produz uma terceira função que representa a quantidade de sobreposição entre h e uma versão invertida e transladada de x.

$$y[n] = x[n] * h[n] = \sum_k x[k] \cdot h[n-k] \tag{10}$$

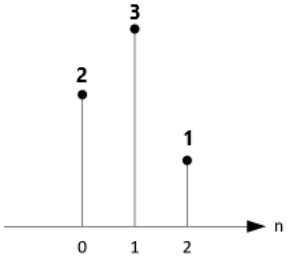
Onde x[n] é o sinal de entrada, h[n] é a resposta ao impulso e y[n] é a saída.

Para formalizar a convolução discreta definida acima lançaremos mão de dois conceitos:

- Decomposição de sinais (especificamente em sinais impulso/delta)
- Resposta impulsional (enquanto saída resultante de um sistema que recebeu um impulso como entrada).

No processo de decomposição, o sinal de entrada é decomposto em componentes aditivas e a resposta resultante do sistema na saída é a soma destas componentes passantes, ou seja, o sinal é desconjuntado numa soma ponderada de sinais básicos. Vejamos pictoricamente:

Figura 16: Decomposição de um sinal em um conjunto de impulsos



$$x[0] = x[0] \cdot \delta[n] \qquad = 2 \cdot \delta[n-0]$$
$$x[1] = x[1] \cdot \delta[n-1] = 3 \cdot \delta[n-1]$$
$$x[2] = x[2] \cdot \delta[n-2] = 1 \cdot \delta[n-2]$$

Fonte: Elaborado pelos autores, 2020.

Da figura, pode-se escrever: $x[n] = x[0] \cdot \delta[n-0] + x[1] \cdot \delta[n-1] + x[2] \cdot \delta[n-2]$.

E, em geral, um sinal pode ser escrito como a soma de impulsos escalonados e deslocados:

$$x[n] = \sum_k x[k] \cdot \delta[n-k] \tag{11}$$

A segunda definição importante nesta formalização é a de resposta impulsional, que é a saída resultante de um sistema que recebeu um impulso como entrada, e é denotado por h[n].

Figura 17: Resposta impulsional

Fonte: Elaborado pelos autores, 2020.

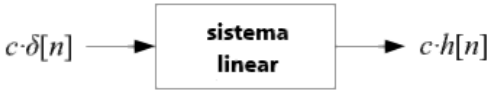
Se o sistema for invariante no tempo (TI) a resposta de um impulso deslocado no tempo também é deslocada na mesma quantidade.

Figura 18: Resposta impulsional - sistema invariante no tempo

Fonte: Elaborado pelos autores, 2020.

Se o sistema for linear, um escalonamento no sinal de entrada causa um escalonamento idêntico no sinal de saída.

Figura 19: Resposta impulsional - sistema linear



Fonte: Elaborado pelos autores, 2020.

Destarte, combinando as propriedades da resposta ao impulso e a idea da decomposição de sinais (via impulsos), nos possibilita construir a equação da convolução discreta. Em um sistema linear e invariante no tempo, a resposta resultante de várias entradas pode ser calculada como a soma das respostas de cada entrada agindo isoladamente:

Figura 20: Resposta impulsional - sistema LTI

Fonte: Elaborado pelos autores, 2020.

Portanto, se o sinal de entrada é $x[n] = \sum_k x[k] \cdot \delta[n-k]$ então a saída será $y[n] = \sum_k x[k] \cdot h[n-k]$.

Notar que a convolução funciona em sistemas lineares e invariantes no tempo, onde um um sinal é decomposto em um conjunto de impulsos e o sinal de saída pode ser calculado adicionando as respostas impulsionais escalonadas e deslocadas.

A elegância matemática reside no fato de que a única característica que precisamos saber sobre o sistema é a sua resposta ao impulso h[n]. Se conhecermos a resposta impulsional do sistema, poderemos descobrir facilmente como o sistema reage a qualquer sinal de entrada.

Referências

[1] WHITTAKER, Edmund Taylor. XVIII.—On the functions which are represented by the expansions of the interpolation-theory. Proceedings of the Royal Society of Edinburgh, v. 35, p. 181-194, 1915

[2] SHANNON, Claude E. Communication in the presence of noise. Proceedings of the IEEE, v. 72, n. 9, p. 1192-1201, 1984.

[3] NYQUIST, Harry. Certain topics in telegraph transmission theory. Transactions of the American Institute of Electrical Engineers, v. 47, n. 2, p. 617-644, 1928.

[4] LATHI, Bhagwandas Pannalal. Sinais e Sistemas Lineares-2. Bookman, 2006.

[5] LATHI, B. P.; DING, Zhi. Sistemas de Comunicações Analógicos e Digitais Modernos-4a edição. Editora LCT, 2012.

[6] COOLEY, James W.; TUKEY, John W. An algorithm for the machine calculation of complex Fourier series. Mathematics of computation, v. 19, n. 90, p. 297-301, 1965.

[7] HAYKIN, Simon. Communication systems. John Wiley Sons, 2008.

[8] JONES, E.; OLIPHANT, E.; PETERSON, P. Scipy: Open Source Scientific Tools for Python [Internet].[Acesso em 03 de Setembro de 2020].