Universidade Federal do Ceará
Centro de Tecnologia
Departamento de Engenharia de Teleinformática
Engenharia de Telecomunicações

Amostragem e Reconstrução de Sinais

Interpolação *Sinc*

Aluno	Lucas de Souza Abdalah - 385472
Professores	João César Moura Mota e André Lima Ferrer de Almeida
Disciplina	Processamento Digital de Sinais - TI0119

Fortaleza, 18 de maio de 2018

Conteúdo

1 Introdução **1**
1.1 Amostragem 1
1.1.1 Trem de Impulsos 1
1.1.2 Amostragem por Trem de Impulsos Periódicos 1
1.2 Teorema de Nyquist 2
1.2.1 Subamostragem e Consequências: *Aliasing* 4
1.3 Reconstrução de Sinal de Banda Limitada 4
1.3.1 Filtragem na Frequência 4
1.3.2 Interpolação por Sinc 6

2 Desenvolvimento **7**
2.1 Problema Proposto 7
2.2 Implementação em Python 3.X 7

3 Análise dos Resultados **10**
3.1 Modelagem 10
3.2 Fenômeno de Gibbs 10

4 Referências **11**

1 Introdução

1.1 Amostragem

1.1.1 Trem de Impulsos

O que norteia o seguinte trabalho é o teorema da amostragem e para desenvolvê-lo, representar amostras de um sinal de tempo conínuo em intervalos uniformemente espaçados (1) é extremamente importante. A função impulso é aplicada em sua representação periódica (2) é utilizada.

$$x[n] = x_c(nT), \quad -\infty < n < \infty \tag{1}$$

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{2}$$

1.1.2 Amostragem por Trem de Impulsos Periódicos

É utilizada a propriedade de amostragem da função, $x(t)\delta(t) = x(0)\delta(t)$. Logo, a técnica "captura"essas amostras ao efetuar o produto entre o sinal $x_c(t)$ de tempo contínuo pelo Trem de Impulsos periódicos $s(t)$

$$x_s(t) = x_c(t)s(t), \tag{3}$$

A equação geral adquire essa forma ao explicitar os termos

$$x_s(t) = x_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = \sum_{n=-\infty}^{\infty} x_c(t)\delta(t - nT) \tag{4}$$

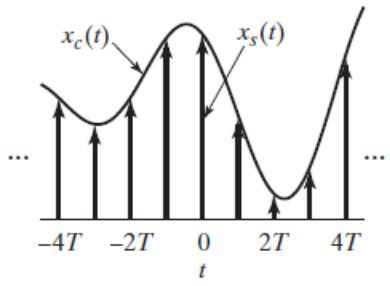

Figura 1: Amostragem do sinal $x_c(t)$ com trem de impulsos com período T.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

O diagrama da figura 2 descreve a operação da amostragem. A multplicação resulta no sinal de tempo contínuo que representa uma soma infitnita das contribuições do sinal contínuo, espaçadas em T, devido a multiplicação pelo trem de impulsos periódico. Ao visualizar o grande bloco de conversão (*C/D Converter*), há de se atentar à discretização do sinal contínuo observada na equação (1):

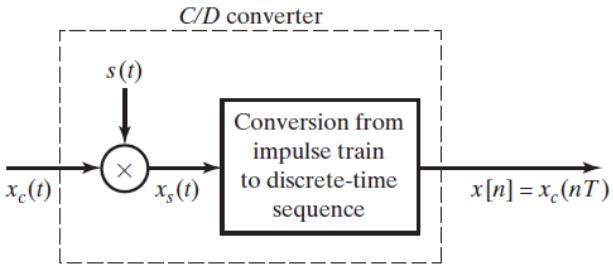

Figura 2: Diagrama de blocos de um conversor de tempo Contínuo-Discreto .
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

1.2 Teorema de Nyquist

Uma análise das proposições de tópicos anteriores no domínio da frequência é essencial para garantir a aplicabilidade do método.

A equação (3) indica que $x_s(t)$ é o produto entre $x_c(t)$ e $s(t)$, e pelas propriedades da transformada de Fourier é sabido que um produto no domínio tempo resulta em uma convulação no domínio da frequência. Além de que a frequencia de amostragem é $\Omega_s = \dfrac{2\pi}{T}$

A transformada de Fourier do impulso periódico:

$$S(j\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s) \tag{5}$$

A convulução dos sinais na frequência:

$$X_s(j\Omega) = \frac{1}{2\pi} X_c(j\Omega) * S(j\Omega) \tag{6}$$

$$X_s(j\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(j(\Omega - k\Omega_s)) \tag{7}$$

O resultado da equação (7) é de importância indiscutível, pois deixa clara a relação de entrada e saída na frequência e que $X_s(j\Omega)$ será composto de cópias de $X_c(j\Omega)$ deslocado por $S(j\Omega)$ e espaçados em Ω_s.

Ao observar o espectro do sinal supondo um $X_c(j\Omega)$, qualquer, e sabido a transformada do impulso periódico, representados abaixo, respectivamente:

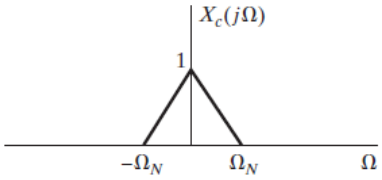

Figura 3: A transformada $X_c(j\Omega)$ do sinal $x_c(t)$.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

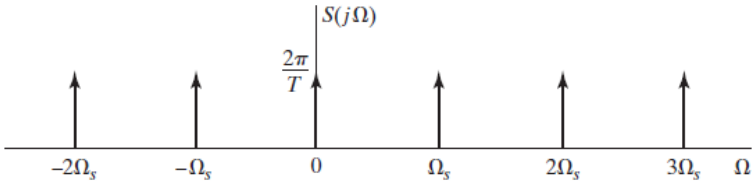

Figura 4: A transformada $S(j\Omega)$ do sinal $s(t)$.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

E ao convolvê-los, tem-se a saída $X_s(j\Omega)$.

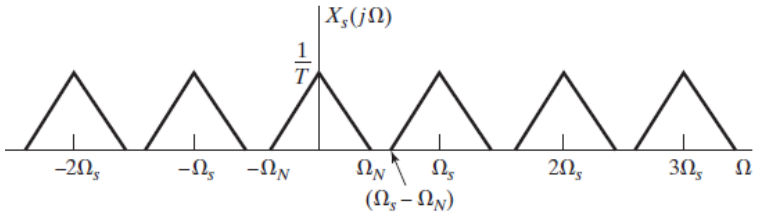

Figura 5: A transformada $X_s(j\Omega)$ do sinal $x_s(t)$.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

Observado o espectro da figura 5, para garantir que não há sobreposição das cópias dos sinais, metade da frequência de amostragem deve ser maior que ou igual a frequência "natural"do sinal amostrado, ficando mais claro com a inequação conhecida como Teorema de Nyquist:

$$\Omega_s - \Omega_n \geq \Omega_n \tag{8}$$

$$\Omega_s \geq 2\Omega_n \tag{9}$$

1.2.1 Subamostragem e Consequências: *Aliasing*

Quando não tomamos o devido cuidado de amostrar de acordo com o proposto no teorema de Nyquist (da amostragem), há a sobreposição das cópias deslocadas no tempo, deixando o sinal irreconhecível, se comparado com o inicial $x_c(t)$. Levando por água abaixo todo o trabalho de amostragem e de conversão feito inicialmente. Esse efeito é conhecido como *Aliasing*.

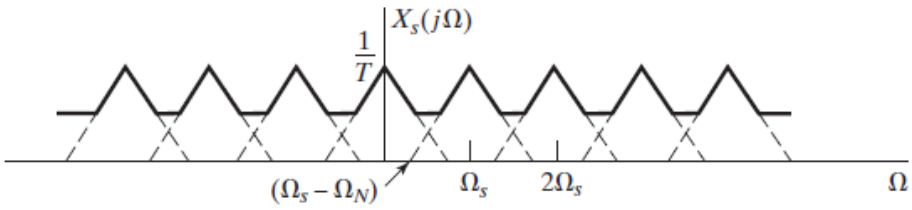

Figura 6: A transformada $X_s(j\Omega)$ quando a não é respeitado o teorema da amostragem. Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

1.3 Reconstrução de Sinal de Banda Limitada

1.3.1 Filtragem na Frequência

Observe que o sinal foi amostrado com suas respectivas cópias. E quando é necessário recuperar apenas o sinal original amostrado?

Podemos aplicar um filto ideal na frequência com sua frequência de corte e seu ganho conhecidos. Teremos a representação abaixo de diagram de blocos.

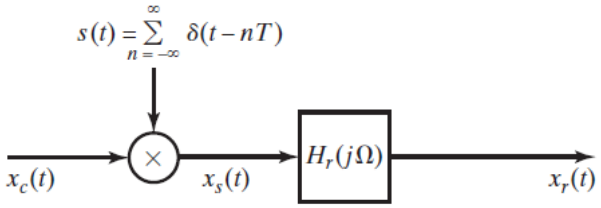

Figura 7: Diagrama de Blocos da aplicação de um filtro ideal na frequência. Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

Representando algebricamente:

$$x_r(t) = h_r(t) * x_s(t) \tag{10}$$

Representação de filtro ideal com ganho T e frequência de corte Ω_c

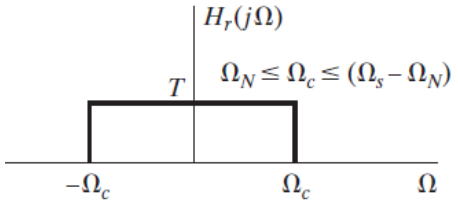

Figura 8: Filtro ideal para recuperação do sinal original.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

As representações seguintes ilustram de forma a ser intuitivo entender a recuperação do sinal e os critérios a serem seguidos.

$$X_r(j\Omega) = H_r(j\Omega)X_s(j\Omega) \tag{11}$$

Visto que o $H_r(j\Omega)$ é o filtro ideal ou a função conhecida como "rect", a sua transformada inversa de Fourier é dada por: $F^{-1}\{H_r(j\Omega)\} = h_r(t)$, sendo essa a conhecida função sinc, um dos temas principais deste trabalho.

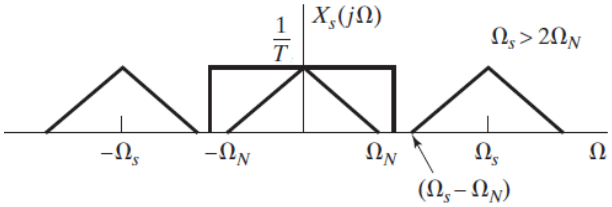

Figura 9: Filtro que recuperando o sinal original.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

Representação da Sinc $h_r(t)$ no domínio do tempo:

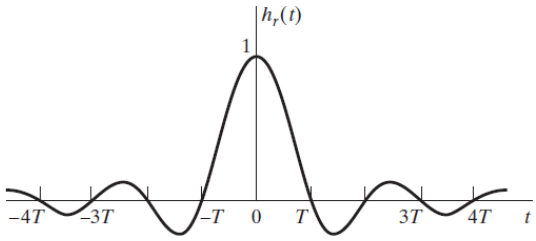

Figura 10: Função sinc de período T.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

E algebricamente pode ser descrita como:

$$h_r(t) = \frac{\Omega_c T sin(\Omega_c t)}{\pi \Omega_c t} \tag{12}$$

1.3.2 Interpolação por Sinc

Levando em consideração toda a carga teórica desenvolvida até este ponto, é possível introduzir o conceito de reconstrução de um sinal de banda limitada com suas amostras.

Expandindo os termos da equação (10) e substituir $h_r(t)$ temos a expressão algébrica (14) e o espectro da figura 11.

$$x_r(t) = \sum_{n=-\infty}^{\infty} x_c(nT) h(t - nT) \tag{13}$$

$$x_r(t) = \sum_{n=-\infty}^{\infty} x_c(nT) \frac{\Omega_c T}{\pi} \frac{sin(\Omega_c(t - nT))}{\Omega_c(t - nT)} \tag{14}$$

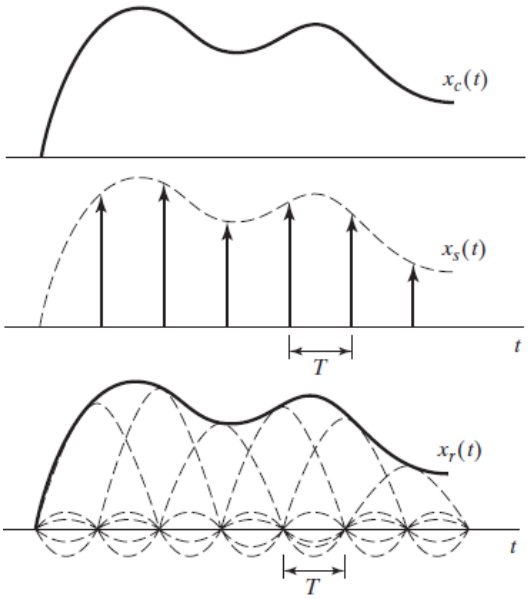

Figura 11: x_c sinal original, x_s sinal amostrado e $x_r(t)$ sinal reconstruido através de interpolação com Sinc, respectivamente.
Fonte: Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009).

2 Desenvolvimento

2.1 Problema Proposto

1. Interpole a sequência $x[n]$ da figura 1 utilizando a função sinc.

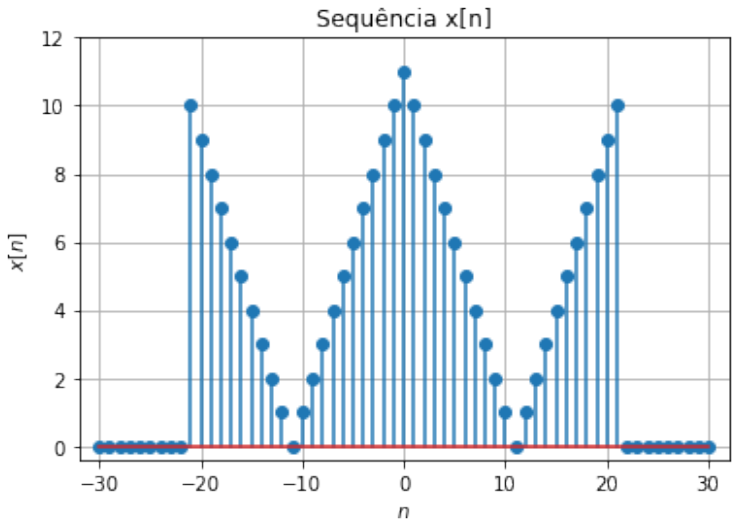

Figura 12: Sequência de amostras original.
Fonte: Elaborada pelo autor.

2.2 Implementação em Python 3.X

Primeiramente, importamos a biblioteca **Numpy** que define diversas funções, operações e constantes matemáticas de uso recorrente, como exponencial (e^x) e a própria função Sinc e geramos a sequência $x[n]$

```
import numpy as np
#Gerando os indices n
n = np.linspace(-30,30,61)
#Gerando as amostras do sinal em si em vetores e np.
    concatenate concatena os conte dos dos vetores
x = np.concatenate((np.zeros(9), np.linspace(10,1,10),
np.array([0]), np.linspace(1,10,10), np.array([11]),
np.linspace(10,1,10), np.array([0]), np.linspace(1,10,10),
np.zeros(9)), 0)
```

Para construir a interpolação, esta função recebe o vetor de amostras de um sinal $x[n]$ e o vetor da sua indexação no tempo $[n]$. Sendo o espaçamento entre as amostras uniforme, calculamos o período utilizando as duas primeiras amostras. Observe que para garantir esse 3º passo, o vetor precisa receber no mínimo 3 amostras de $x[n]$, pois o comparador utiliza essas três para tentar garantir o espaçamento uniforme. Dentro dos dois laços, cada posição do vetor auxinterpolador é preenchido pela contribuição da sinc nessa posição

```
def interpolador(x, n):
    #Calculamos o espacamento(periodo) entre as amostras
    Ts = (n[1]- n[0])

    #Inicia a verificacao do espacamento das amostras.
    if (Ts != n[2] - n[1]):
        return("Espacamento nao uniforme.")

    else:
        #Utiliza a dimensao do espaco de amostras n para
            estimar um espaco de tempo continuo
        halfsizen = int(len(n)/2)

        #Estima um tempo continuo (com o passo(distancia
            entre as amostras) muito menor que em n)
        t = np.linspace(-halfsizen+1,halfsizen,20*(halfsizen)
            )

        #vetor para inicializar a interpolacao, variavel que
            tem metade do tamanho do vetor n
        auxinterpolador = np.zeros(len(t))

        #Laco que prepara o ambiente para preencher o vetor
            auxinterpolador
        for indice in range(0,len(t)):
            aux=0
            #Laco que realmente preeche o vetor recuperando
                as amostras de um sinal
            #obs: observe que o laco precisa percorrer os
                valores positivos e negativos do dominio
            for k in range(-halfsizen,halfsizen,1):
                auxinterpolador[indice] += (x[aux]*(np.sinc
                    (((t[indice])/Ts)-k)))
                aux += 1

        #retorna o novo espaco de amostras e estima o sinal
            original
        return(auxinterpolador, t)
```

E assim, o retorno da função é a própria reconstrução do sinal e o tempo contínuo.

```
import matplotlib.pyplot as plt
plt.stem(n,x)
plt.title('Sequencia x[n]')
plt.figure('1')
#Calcula a interpolacao com a funcao
a = interpolador(x,n)
#Atribuicao do retorno da funcao
aint, at = a[0], a[1]
#Plota o sinal estimado contra o tempo continuo
plt.plot(at,aint, 'r')
#Sinal discretizado em n
plt.stem(n,x)

#Formata o do gr fico
plt.title('Reconstru o do Sinal x(t) com a sequ ncia x[n]
    ')
plt.xlabel(r'$n$')
plt.ylabel(r'$x[n]$')
plt.axis([-32,32,-0.4,12])
plt.grid('on')
plt.figure('2')
plt.show()
```

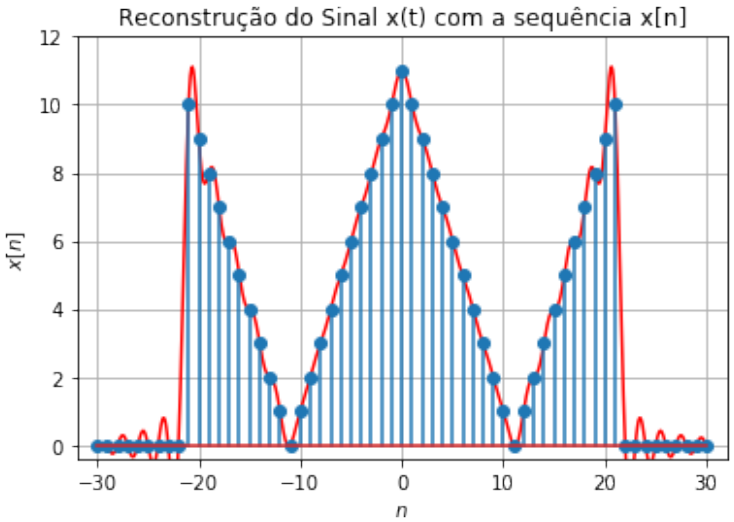

Figura 13: Reconstrução do Sinal e sequência de amostras original.
Fonte: Elaborada pelo autor.

3 Análise dos Resultados

3.1 Modelagem

Quando reconstruímos um sinal por interpolação com Sinc, teoricamente, teremos o sinal amostrado exato, pois temos a contribuição dos infinitos termos da Sinc. Além, é óbvio de respeitar o teorema da amostragem.

No caso da sequência analisada nesse trabalho, apesar da limitação de recursos de memória e de processamento dos sistemas, que impossibilita efetuar uma "soma infinita"de forma a obter o sinal exato, foram utilizadas contribuições grandes o suficiente para termos uma representação muito aproximada do sinal original.

E por que tivemos distorções na reconstrução mesmo se tivermos uma contribuição muito grande das sincs?

A quantidade de amostras do sinal foi insuficiente, ou seja, a mensagem sofreu um processo de subamostragem, consequentemente ao efetuar a reconstrução com o filtro projetado, houve sobreposição das cópias no espectro da frequência e uma distorção no tempo. Como não temos controle no processo de amostragem, apenas recebemos a mensagem, não temos como aumentar o número de amostras e diminuir o período de amostragem. Sendo isso possível, poderíamos aumentar suficientemente o número de amostras para um reconstrução muito próxima da original.

Porque não podemos ter o sinal perfeito?

Pois não podemos fazer uma a operação de somatórios infinitos. Mas temos como estimar esse sinal computacionalmente, pois os sistemas e os *softwares* atuais, têm um "absurdo"poder de processamento e capacidade de efetuar milhares de cálculos em poucos segundos. Então, em vez de tentarmos uma soma infinita, somamos um número muito grande de parcelas, para nos aproximarmos cada vez mais do sinal calculado teoricamente.

Fazendo o número de somas das parcelas, vemos o sinal convergindo para o estimado, porém com algumas pequenas variações.

3.2 Fenômeno de Gibbs

O filtro ideal é irrealizável, a transformada inversa de Fourier dele tampouco. As perturbações associadas a regiões de descontinuidade e de mudanças muito abruptas, decorrem também do Fenômeno de Gibbs. Entretanto, em regiões que apresentam simetria o resultado é um sinal mais comportado. Com maior frequência de amostragem, há oscilações mais contidas e sinal "comportado", porém o fenômeno ocorrerá independendo do quão grande for a f_s, pois sendo $f_s < \infty$, e o efeito ocorre, por menor que seja.

4 Referências

[1] Oppenheim, Alan V. – Signals and systems, 2nd Edition;

[2] Oppenheim, A. V., and Schafer, R. W., Discrete-Time Signal Processing - Prentice Hall (2009);

[3] Time Domain Sinc Interpolation Resampling;
mathworks.com/matlabcentral/fileexchange/59027-time-domain-sinc-interpolation–resampling- Acesso: 17/05/2018 às 15h30

[4] Nyquist Interpolation – Nicholas Work ;
nicholasdwork.com/tutorials/nyquistInterp.pdf Acesso: 17/05/2018 às 17h10

[5] Whittaker–Shannon interpolation formula;
en.wikipedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula
Acesso: 18/05/2018 às 13h10

[6] Perfect Sinc Interpolation;
gist.github.com/endolith/1297227 Acesso: 15/05/2018 às 12h30