2o e 3o Trabalhos de DSP

Prof. Dr. Ricardo J. Ferrari

10 de Outubro de 2021

2o Trabalho (data máxima de entrega 17/10)

- a) Estude o material disponível no site abaixo e gere um resumo (em português) de 2 a 4 páginas do que vocês entenderam sobre o material - foque, e deixe claro no resumo, as diferenças entre convolução cíclica (ou circular) e acíclica.
- https://www.dsprelated.com/freebooks/sasp/Convolution_Short_Signals.html
- b) Estude os exemplos em Octave e implemente-os em python3. **Os exemplos em python deverão ser realizados usando o jupyter notebook** [4] - o grupo deverá submeter via moodle apenas o arquivo do resumo (.doc, .ipynb, .lyx ou .pdf) e o "arquivo ipynb". O resumo poderá ser integrado juntamente com os exemplos python em um único arquivo do tipo jupyter notebook (.ipynb). Nesse caso, o grupo poderá submeter no ava apenas o arquivo .ipynb.

OBS: Resumos no estilo "copy -> Google Translate -> paste" receberão nota ZERO.

3o Trabalho (data máxima de entrega 14/11)

Neste projeto você estudará a filtragem por métodos de convolução. Todas as implementações devem ser realizadas usando as **linguagens python3**.

Antes de realizar a implementação dos métodos, é altamente recomendado revisar o material de estudo do 2o Projeto - https://www.dsprelated.com/freebooks/sasp/Convolution_Short_Signals.html

A avaliação levará em consideração não só apenas se os métodos implementados estão funcionando corretamente, mas também a qualidade da documentação do código e das discussões dos resultados. **O projeto deverá ser realizado usando o jupyter notebook** [4] - o grupo deverá submeter via moodle apenas o "arquivo ipynb".

1 Convolução

A convolução de um filtro, h_n, $n = 0, 1, \ldots, M$, de ordem M, e um sinal causal de duração finita, x_n, $n = 0, 1, \ldots, L-1$, de comprimento L, é definida como:

$$y_n = \sum_{m=\max(0,n-L+1)}^{\min(n,M)} h_m x_{n-m}, \quad n = 0, 1, \ldots, L+M-1. \tag{1}$$

2 Prática

Considere um filtro semelhante a um integrador definido pela equação de Entrada/Saída:

$$y(n) = 0,1 \times [x(n) + x(n-1) + x(n-2) + \cdots + x(n-14)]\,. \tag{2}$$

Esse filtro acumula (integra) as 14 amostras do sinal de entrada, ou seja, a atual e mais 13 anteriores. O fator $0,1$ representa apenas um fator de escala conveniente para este experimento. Segue-se que a resposta ao impulso deste filtro é dada por

$$h_n = \begin{cases} 0,1 & \text{para } 0 \leq n \leq 14 \\ 0 & \text{caso contrário} \end{cases}. \tag{3}$$

2.1 Atividades:

a) Escreva uma função python3 de nome **conv1** que implementa a Equação 1. Ela deva ser usada como

$$y = \text{myconv1}(x, h), \tag{4}$$

em que x, h e y são os vetores representando o sinal de entrada, o filtro e o sinal de saída, respectivamente. A função deve ser capaz de aceitar como entrada os vetores x e h, se eles forem inseridos como linhas ou colunas. Além disso, o vetor de saída e deve corresponder ao mesmo tipo do vetor de entrada x, ou seja, linha ou coluna.

b) Para observar a resposta de estado estacionário do filtro descrito pela Equação 3, bem como os transientes de entrada e saída da entrada, considere um sinal de entrada dado por uma onda quadrada x_n de comprimento $L = 200$ e período de $K = 50$ amostras. Na linguagem C, tal sinal pode ser obtido pelo simples Algoritmo 1:

Algoritmo 1 Algoritmo em C para gerar uma onda quadrada de comprimento L e período K.

```
for (n=0; n<L; n++)

    if (n%K < K/2) /* n%K is the MOD operation */
        x[n] = 1;
    else
        x[n] = 0;
```

c) Usando sua função **conv1**, calcule o sinal de saída y_n e plote-o no mesmo gráfico com x_n(x_n versus n e y_n versus n) . À medida que a onda quadrada liga e desliga periodicamente, **verifique e relate** os comportamentos "transiente ligado", "estável" e "transiente desligado" do filtro.

d) Verifique (**plote para comparação**) se sua função produz os mesmos resultados que a função em python3 "numpy.convolve".

e) Usando a relação entre convolução no espaço e no domínio da frequência dada pela Equação 5

$$x_n * h_n = ifft\left[fft(x_n) \cdot fft(h_n)\right], \tag{5}$$

sendo fft e $ifft$, respectivamente, as transformadas rápidas de Fourier direta e inversa, implemente uma função de nome **conv2** (com os mesmos parâmetros de entrada e saída da função **conv1**) que faça a convolução dos sinais x_n e h_n no domínio da frequência (lado direito da Equação 5). Observe que a função $ifft$ retorna valores complexos cuja parte imaginária não é exatamente zero como deveria ser devido à precisão numérica finita; nesse caso, considere apenas a parte real dos números. Na implementação da função **conv2** use padding/cropping, caso os sinais tenham tamanhos diferentes. Não é aceitável o uso de funções prontas para a convolução, a não ser quando especificado.

f) Usando agora a nova função de convolução **conv2**, repita as análises realizadas com a função **conv1** (itens c e d).

g) Repita os itens c e d para o filtro

$$h_n = \begin{cases} 0,25 \times (0,75)^n & \text{para } 0 \leq n \leq 14 \\ 0 & \text{caso contrário} \end{cases}, \tag{6}$$

que age mais como um filtro do tipo integrador-RC (resistor-capacitor) do que um acumulador. Este filtro atua como um diferenciador. Assim, em seu estado estacionário, ele diferencia uma constante para zero. Conforme a onda quadrada assume valores 0 e 1, você poderá observar essa ação diferenciadora do filtro, bem como os transientes gerados.

h) Repita os itens c e d para o filtro cuja função de transferência é

$$H(z) = \frac{1}{5}\left(1 - z^{-1}\right)^5. \tag{7}$$

i) Para demonstrar os conceitos de resposta ao impulso, linearidade e invariância no tempo, considere um filtro com resposta ao impulso finita $h_n = (0,95)^n$, para $0 \leq n \leq 24$. O sinal de entrada, $x(n) = \delta(n) + 2\delta(n-40) + 2\delta(n-70) + \delta(n-80)$, $n = 0, 1, \ldots, 120$, consiste em quatro impulsos de intensidades indicadas que ocorrem nas instâncias de tempo indicadas. Observe que os dois primeiros impulsos são separados por uma duração maior que a duração do filtro, enquanto os dois últimos são separados por menos. Usando sua função **conv1**, calcule a saída do filtro y_n para $0 \leq n \leq 120$ e plote-a no mesmo gráfico com x_n. Comente sobre a saída resultante em relação à linearidade e invariância no tempo. Ao gerar o sinal de entrada acima, você pode achar útil a seguinte função delta escrita em C:

Algoritmo 2 Algoritmo em C para gerar uma função delta, δ.

```
/* delta.c - delta function */
double delta(n)
int n;
{

    if (n == 0)
        return 1;
    else
        return 0;

}
```

OBS1: Ótimo material de consulta - https://greenteapress.com/thinkdsp/html/index.html

Referências

[1] https://www.dspguide.com/ch18/2.htm

[2] https://www.dsprelated.com/freebooks/sasp/Convolution_Short_Signals.html

[3] https://octave.org/doc/v4.2.1/Signal-Processing.html

[4] https://medium.com/horadecodar/como-instalar-o-jupyter-notebook-windows-e-linux-20701fc583c