Prof. Fernando Passold
Eng. Elétrica
Lab de Processamento de Sinais
7 de setembro de 2023

Lab 2) Convolução e Reverberação

Embasamento teórico

Convolução continua no tempo

A convolução contínua no tempo é uma operação matemática que combina duas funções para criar uma terceira função. Ela é frequentemente usada em sistemas lineares e invariáveis no tempo, como sistemas de circuitos elétricos, processamento de sinais e engenharia de controle.

A convolução de duas funções $f(t)$ e $g(t)$ é denotada como:

$$(f * g)(t)$$

e é definida da seguinte forma:

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

Convolução discreta no tempo

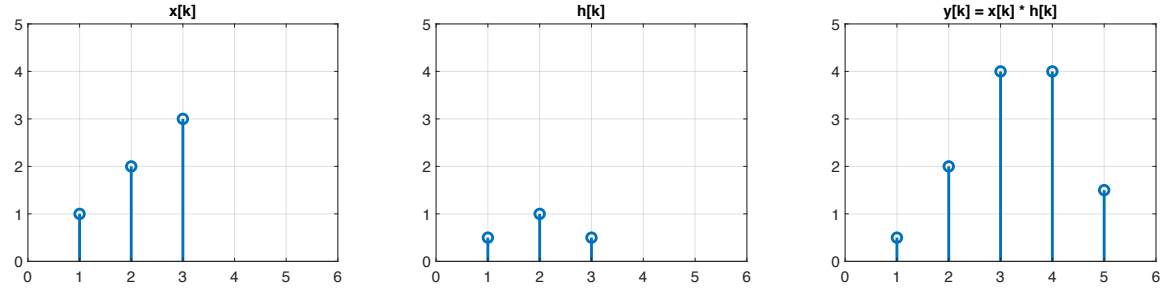
A convolução discreta no tempo é semelhante à convolução contínua, mas é aplicada a sinais discretos. Em vez de integração, utiliza somatórios. Se tivermos duas sequências discretas $x[k]$ e $h[k]$, a convolução discreta $y[k]$ é dada por:

$$y[k] = \sum_{n=-\infty}^{\infty} x[n] \cdot h[n - k]$$

Exemplo de Convolução Discreta usando MATLAB:

Suponha 2 sequencias discretas simples: $x[k] = [0,1,2,1,0]$ e $h[k] = [0,1,1,0]$.

A convolução discreta destes 2 sinais resulta:



Este resultado foi obtido usando-se Matlab:

```
% convol1.m
% Fernando Passold, em 29/08/2023
% Exemplo simples de convoluÃ§Ã£o discreta

% SequÃªncias de entrada
x = [1, 2, 3];
h = [0.5, 1, 0.5];

% Comprimento das sequÃªncias
len_x = length(x);
len_h = length(h);

% InicializaÃ§Ã£o da saÃda da convoluÃ§Ã£o
y = zeros(1, len_x + len_h - 1);

% CÃ¡lculo da convoluÃ§Ã£o
for n = 1:length(y)
    for k = 1:len_h
        if n - k + 1 > 0 && n - k + 1 <= len_x
            y(n) = y(n) + x(n - k + 1) * h(k);
        end
    end
end

disp("Resultado da convoluÃ§Ã£o y[n]:");
disp(y);

% plotando os resultados para melhor compreensÃ£o
subplot(131); stem(x)
axis([0 6 0 5])
grid
title('x[k]')

subplot(132); stem(h)
axis([0 6 0 5])
grid
title('h[k]')

subplot(133); stem(y)
axis([0 6 0 5])
grid
title('y[k] = x[k] * h[k]')
```

Note: uma convolução discreta em tempo-real com um sinal de áudio pode ser usado para modelar as características de sistemas acústicos ou para aplicar efeitos sonoros complexos.

Um exemplo clássico é a simulação de reverberação ("*reverb*"), onda a convolução é usada para simular o som refletido em um ambiente específico.

Para realizar uma convolução discreta em tempo-real com um sina de áudio, pode-se seguir os seguintes passos:

1. **Preparação**: Tenha o sinal de áudio que você deseja convolver (a entrada) e a resposta ao impulso do sistema que você deseja modelar (o filtro de convolução). A resposta ao impulso é uma representação da maneira como o sistema responde a um impulso unitário.
2. **Amostragem**: Ambos os sinais, o sinal de áudio e a resposta ao impulso, precisam ser discretizados, pois você está trabalhando com uma convolução discreta. Eles devem estar alinhados adequadamente em termos de amostragem e tamanho.
3. **Convolução Discreta**: Aplique a convolução discreta entre o sinal de áudio e a resposta ao impulso usando um laço de soma ponderada, como mostrado no exemplo de código anterior.
4. **Saída**: O sinal de saída da convolução representa como o sinal de áudio original seria alterado ao passar pelo sistema modelado pela resposta ao impulso. Isso pode resultar em efeitos como reverb, eco, filtragem, etc.
5. **Streaming em Tempo Real**: No contexto de tempo real, você realizará a convolução continuamente enquanto recebe novos dados de entrada. Isso pode ser feito em blocos de amostras para processamento eficiente.

No MATLAB, você pode usar a função `conv` para realizar a convolução de forma eficiente. Aqui está um exemplo simplificado de como você poderia aplicar a convolução em tempo real a um sinal de áudio:

```
% Carregar um sinal de áudio e uma resposta ao impulso
audio_signal = audioread('audio_file.wav');
impulse_response = audioread('impulse_response.wav');

% Realizar a convolução em tempo real
output_signal = conv(audio_signal, impulse_response);

% Reproduzir o sinal de saída
sound(output_signal, Fs);  % 'Fs' é a taxa de amostragem do áudio
```

Uma versão melhorada será disponibilizada como "`convol2.m`".

Respostas ao Impulso (IR)

Há várias fontes onde você pode encontrar arquivos de resposta ao impulso (IR) para ambientes como catedrais. Esses arquivos são frequentemente usados para simular a acústica desses ambientes em aplicações de processamento de áudio e produção musical. Aqui estão algumas opções:

1. **OpenAIR**: O Open Acoustic Impulse Response Library (OpenAIR) é uma biblioteca de arquivos IR de ambientes diversos, incluindo catedrais disponibilizado por Adam Townsell. Se oferecem uma ampla gama de IRs gratuitos para download. Visite o site: http://www.openairlib.net/
2. **SIR Reverb**: O SIR (Super Impulse Response) é um programa gratuito que permite criar seus próprios arquivos de resposta ao impulso ou usar os disponíveis na comunidade. Muitos deles incluem ambientes como catedrais. Site: http://www.knufinke.de/sir/sir1.html
3. **EchoThief:** O EchoThief é um site onde você pode encontrar uma variedade de arquivos IR, incluindo ambientes como catedrais. Eles oferecem algumas amostras gratuitas e também opções para compra. Site: https://www.echothief.com/ ou http://www.echothief.com/downloads/ ou https://sites.google.com/sdsu.edu/echothief
4. **ConvolvIR**: ConvolvIR é um site que oferece arquivos IR de alta qualidade para simulação de ambientes. Eles têm uma coleção diversificada que inclui catedrais. Site: http://www.convolvir.com/
5. **Pulse Exploration**: O Pulse Exploration é um projeto que disponibiliza arquivos IR gratuitos de locais reverberantes, como igrejas e catedrais. Site: http://www.pulseexploration.com/
6. **SampleSwap**: O SampleSwap é uma comunidade de compartilhamento de samples e oferece algumas respostas ao impulso para ambientes reais. Site: https://sampleswap.org/
7. **Bibliotecas de Plugins de Reverb:** Muitos plugins de reverb incluem uma variedade de IRs, incluindo catedrais. Alguns exemplos são o "Altiverb" e o "IR1" da Waves.

Lembre-se de verificar os termos de uso e as restrições ao usar arquivos IR de diferentes fontes. Além disso, ao usar esses arquivos em seus projetos, é importante ajustar a IR ao seu ambiente virtual de acordo com a taxa de amostragem e as configurações de duração.

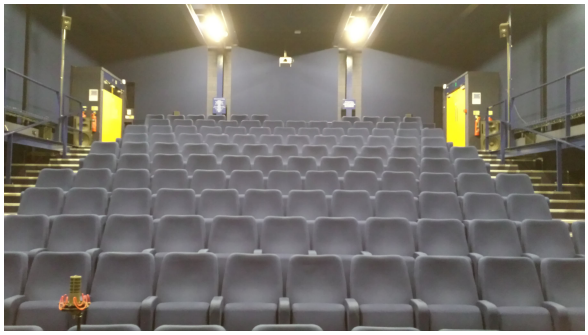
Exemplo de Uso

Usando a rotina `convol2.m` testando diferentes resultados obtidos usando-se fontes de sinais extraídos do repositório OpenAIR:

```
>> dir *.wav

25t_ir_wgw.wav                bvf-piece.wav                 r1_bformat-48k.wav
Batcave.wav                   classichorn.wav               r1_omni_48k.wav
CathedralRoom.wav             courtyard_recorded_ir.wav     s1r2.wav
MLKDream.wav                  drums.wav                     s1r7.wav
adult_female_speech.wav       flute_music.wav               singing.wav
baptist_nashville_balcony.wav output_signal.wav             stairwell_ortf.wav
baptist_nashville_far_wide.wav paphorn.wav

>> convol2
Teste de convolução (reverberação) de sinais

Informe nome arquivo audio de entrada ? singing.wav
Freq. de amostragem adotada: 48.00 [KHz]

Informe nome arquivo audio de IR ? s1r7.wav
Freq. de amostragem adotada: 96.00 [KHz]

Sinal IR possui 4 canais. Convertendo para mono...

ATENÇÃO: Freq. de amostragem do sinal de entrada
é diferente do sinal de IR
Reduzindo Sampling rate do sinal IR
Para nova taxa de amostragem: 48.00 [KHz]

Realizando convolução...
521872 amostras serão geradas (10.87 segundos).
Aguarde...
Normalizando amplitudes vetor de saída...

Gravado arquivo <<output_signal.wav>>...
Para repetir saída gerada, digite:
       sound(output_signal, menor_Fs);

>>
```

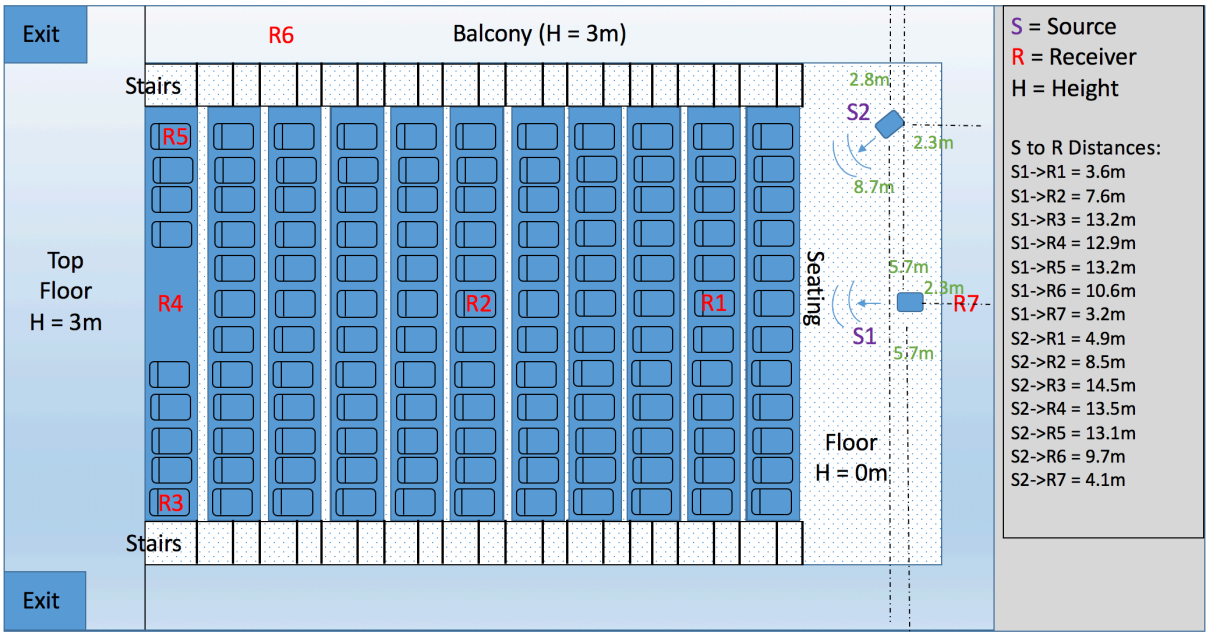
Note que **`s1r7.wav`** é a resposta ao impulso que seria obtida se um ouvinte se sentasse na posição do auditório da Universidade de York (procure por "`arthur-sykes-rymer-auditorium-university-york`" no repositório da **OpenAir**) mostrado na próxima figura.

Fig.: Auditório da Universidade de York (repositório OpenAir).



(a) Visão à partir do palco.

(B) Visão à partir do seu fundo



(c) Planta baixa do auditório

Exercício:

Apresente 3 simulações, mesclando o mesmo sinal de entrada $x[k]$ (sinal de áudio sem processamento) por 3 sinais diferentes $h[k]$ (respostas ao impulso para diferentes ambientes)

Fim.

Anexo

Código `convol2.m`:

```
% convol2.m
% Fernando Passold, em 31/08/2023
clear x h x_mono h_mono

%% Carregar um sinal de Audio e uma resposta ao impulso
disp('Teste de convolucao (reverberacao) de sinais')
disp(' ');
file_x = input('Informe nome arquivo audio de entrada ? ', 's');
% audio_signal = audioread('audio_file.wav');
[x, Fs_x] = audioread(file_x);
fprintf('Freq. de amostragem adotada: %.2f [KHz]\n\n', Fs_x/1000);

%% Eventualmente o arquivo de entrada Ã© stereo (2 canais)
% se faz necessÃirio converter de stereo --> mono
% Ref.: https://la.mathworks.com/matlabcentral/answers/345155-how-to-check-number-of-
channels-of-a-sound-file-and-convert-stereo-file-in-mono-in-matlab
[L_x, n_x] = size(x); % n_x = numero de canais
if n_x >= 2
    % trabalha apenas com os 2 primeiros canais, supondo sinal stereo
    fprintf('Sinal de entrada possui %i canais. Convertendo para mono...\n\n', n_x);
    x_mono = x(:, 1) + x(:, 2);
    peakAmp = max(abs(x_mono));
    x_mono = x_mono/peakAmp;
    %  check the L/R channels for orig. peak Amplitudes
    peakL = max(abs(x(:, 1)));
    peakR = max(abs(x(:, 2)));
    maxPeak = max([peakL peakR]);
    % apply x's original peak amplitude to the normalized mono mixdown
    x_mono = x_mono*maxPeak;
else
    x_mono = x;
end

%% Carrega arquivo IR
file_h = input('Informe nome arquivo audio de IR ? ', 's');
% impulse_response = audioread('impulse_response.wav');
% impulse_response = audioread(file_h);
[h, Fs_h] = audioread(file_h);
fprintf('Freq. de amostragem adotada: %.2f [KHz]\n\n', Fs_h/1000);

%% Eventualmente o arquivo de entrada Ã© stereo (2 canais)
% se faz necessÃirio converter de stereo --> mono
% Ref.: https://la.mathworks.com/matlabcentral/answers/345155-how-to-check-number-of-
channels-of-a-sound-file-and-convert-stereo-file-in-mono-in-matlab
[L_h, n_h] = size(h); % n_h = numero de canais
if n_h >= 2
    % trabalha apenas com os 2 primeiros canais, supondo sinal stereo
    fprintf('Sinal IR possui %i canais. Convertendo para mono...\n\n', n_h);
    h_mono = h(:, 1) + h(:, 2);
    peakAmp = max(abs(h_mono));
    h_mono = h_mono/peakAmp;
    %  check the L/R channels for orig. peak Amplitudes
    peakL = max(abs(h(:, 1)));
    peakR = max(abs(h(:, 2)));
    maxPeak = max([peakL peakR]);
    % apply x's original peak amplitude to the normalized mono mixdown
    h_mono = h_mono*maxPeak;
else
    h_mono = h;
end

%% Compatibiliza Fs's se diferentes...
if Fs_x ~= Fs_h
    fprintf('ATENÃ‡ÃƒO: Freq. de amostragem do sinal de entrada\n');
    disp('Ã© diferente do sinal de IR');
    fprintf('Reduzindo Sampling rate do sinal ');
    menor_Fs = min([Fs_x Fs_h]);
    maior_Fs = max([Fs_x Fs_h]);
    % Ref.: https://la.mathworks.com/help/signal/ug/changing-signal-sample-rate.html#
    [P, Q] = rat(maior_Fs/menor_Fs);
    % descobre qual sinal deve ser re-sampleado
    if maior_Fs == Fs_h
```

```
        disp('IR');
        h_mono = resample(h_mono, P, Q);
        [L_h, n_h] = size(h_mono);
    else
        disp('de entrada');
        x_mono = resample(x_mono, P, Q);
        [L_x, n_x] = size(x_mono);
    end
    fprintf('Para nova taxa de amostragem: %.2f [KHz]\n\n', menor_Fs/1000);
end

%% Realizar a convoluÃ§Ã£o em tempo real
% output_signal = conv(audio_signal, impulse_response);
disp('Realizando convoluÃ§Ã£o... ')
L = max([L_x+L_h-1, L_x, L_h]); % estimando qtdade de amostras geradas...
fprintf('%i amostras serÃ£o geradas (%.2f segundos).\nAguarde...\n', L, L/menor_Fs);
output_signal = conv(x_mono, h_mono);

% Normalizando o vetor de saida obtido, senÃ£o ocorre eventualmente:
% Warning: Data clipped when writing file.
% Ocasionado porque:
% For 16 bit precision, the values are limited to â€“1.0 <= y < +1.0,
% when the signal is provided as floating point format.
% A workaround is to convert the data manually before calling wavwrite()

output_signal = output_signal ./ max(abs(output_signal(:)));

%% Reproduzir o sinal de saÃda
disp('Normalizando amplitudes vetor de saÃda...')
sound(output_signal, menor_Fs);  % 'Fs' Ã© a taxa de amostragem do Ãudio

filename = 'output_signal.wav';
audiowrite(filename, output_signal, menor_Fs);
fprintf('\nGravado arquivo <<%s>>...\n', filename);
disp('Para repetir saÃda gerada, digite:');
fprintf('\tsound(output_signal, menor_Fs);\n\n')
```