Table of Contents


```
% Yuval Epstain Ofek
% Sound & Space HW2

clear all;close all;clc;
```

Parameters

```
%Time
T = 16;
%Sampling freq
fs = 1024;
```

Derived constants

```
%samples
N = T*fs;
%delta f
df = fs/N;
```

Generating input data

```
%time series
t = (0:(N-1))/fs;
%input signal frequency
f = 100;
%Generate input signal;
n = 10*randn(1,N);          %noise
x_actual = sin(2*pi*f*t);   %signal
x= x_actual+n;

%fft
[X,f] = my_fft(x,N,fs);
% Xmag = 20*log(abs(X));
% plot(f,Xmag)

%Sxx
Sxx_full = get_Sxx(X,T);
```

```
%Gxx
[Gxx_full,fgxx_full] = get_Gxx(Sxx_full,f, df, N);
Gxx_full_mx = max(max(Gxx_full));
Gxx_full_dB = 10*log(Gxx_full/Gxx_full_mx);
%%MS check
MScheck(Gxx_full, Sxx_full, x, X, df, T, N)
```

Constants for the split-time analysis

```
N_16 = 1024;
df_16 = fs/N_16;

%Initialize variables
X_16 = zeros(16,N_16);
Gxx_16 = zeros(16,513);
Sxx_16 = zeros(16,N_16);

%Gxx's
for i = 1:16
    [X_16(i,:), f_16] = my_fft(x((1:1024)+1024*(i-1)),N_16,fs);
    Sxx_16(i,:) = get_Sxx(X_16(i,:),1);
    [Gxx_16(i,:),fgxx_16] = get_Gxx(Sxx_16(i,:),f_16, df_16, N_16);
end
%average
Gxx_16_mean = mean(Gxx_16,1);
Gxx_16_mx = max(max(Gxx_16));
Gxx_16_dB = 10*log(Gxx_16_mean/Gxx_16_mx);
%MS check (only on the last one - since it's a loop and the same steps
 are
%performed each time, 1 check should suffice to show all are good) -
 the
%output also looks really confusing when put inside the loop
MScheck(Gxx_16(i,:), Sxx_16(i,:), x((1:1024)+1024*(i-1)), X_16(i,:),
 df_16, 1, N_16)



%%%Plotting
figure
plot(fgxx_full, Gxx_full_dB, 'LineWidth', 1.25)
hold on
plot(fgxx_16,Gxx_16_dB, 'LineWidth', 1.25)
title('16 segment average of Gxx')
title('Gxx Plots', 'FontSize', 18, 'FontWeight', 'bold')
xlabel('Frequency (Hz)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Amplitude (dB re: max)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor;
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
xlim([0, 512])
ylim([min(min(Gxx_full_dB))-10, 5])
```

```
legend('Gxx of full time segment', '16 segment Gxx
 average', 'FontSize', 16)
```

Conclusions:

Spread of the noise is reduced, but we also reduce the value of our peak

Functions:

my_fft

```
function [X,f,sum_check_t,sum_check_f] = my_fft(x,N, fs)
%[X,f,sum_check_t,sum_check_f,checkshift] = my_fft(x,N,fs)
%Takes the N point FFT of x and multiplies by 1/fs. Shifts the
 reference
%output and provides the shifted frequencies for plotting. Two outputs
 to
%check if the FT was performed properly, and one more to check if the
 shift
%is like fftshift.

%Make input row vector
x= x(:).';

dt = 1/fs;
df = fs/N;
%Take FT
Xpshift = fft(x,N)*dt;
%Checking if FT was good using Parseval's
sum_check_t= sum(x.^2)*dt;
sum_check_f= (sum(abs(Xpshift).^2))*df;

%shifting X
X = my_fftshift(Xpshift, N);

%Since my shift is like fftshift, the fs/2 component goes to -fs/2,
 which
%should be the first element of f:
f = [-floor(N/2):-1,0:ceil(N/2)-1]*df;
end
```

my_fftshift

```
function X = my_fftshift(X, N)
%X = myfftshift(X,N)
% My interpertation of fftshift. Takes the input and divides it into
 2,
% when the size is odd, we let the make the first half (indices 1 to
% ceil(half)) be the bigger "half", and swap the two sections.
X = [X(ceil(N/2)+1:end), X(1:ceil(N/2))];
end
```

```
function Sxx = get_Sxx (X,T);
%Sxx = get_Sxx (X,T);
% Finds the Sxx of the input X for time T.
    Sxx = X.*X'.'/T;
end

function [Gxx,fgxx] = get_Gxx(Sxx,f, df, N)
%Gxx = get_Gxx(Sxx,f, df, N)
%For a given Sxx, f, df, and N, we produce the Gxx and a frequency
 vector
%that matches it
I0 = find(f==0);
Gxx = [Sxx(I0),Sxx(I0+1:end)*2,Sxx(1)];
fgxx = (0:floor(N/2))*df;
end

function MScheck(Gxx, Sxx, x, X, df, T, N)
% RMScheck(Gxx, Sxx, x, X, df, T, N)
% Prints MS values to check if Gxx, Sxx, and FT were done correctly
MSCGxx = sum(Gxx*df)
MSCSxx = sum(Sxx*df)
MSCxn = 1/N*sum(x.^2)
MSCXm = 1/T*sum(X.*X'.')*df
end
```

```
MSCGxx =

  100.3548
```

```
MSCSxx =

  100.3548
```

```
MSCxn =

  100.3548
```

```
MSCXm =

  100.3548
```

Published with MATLAB® R2018b