Cours : Traitement du signal TP 5 – Sujet 1/2/3/4/5 - Noté 8/1/2026	ISEN ALL IS DIGITAL! BREST yncréa	Houssem Gazzah CIPA4 Nantes

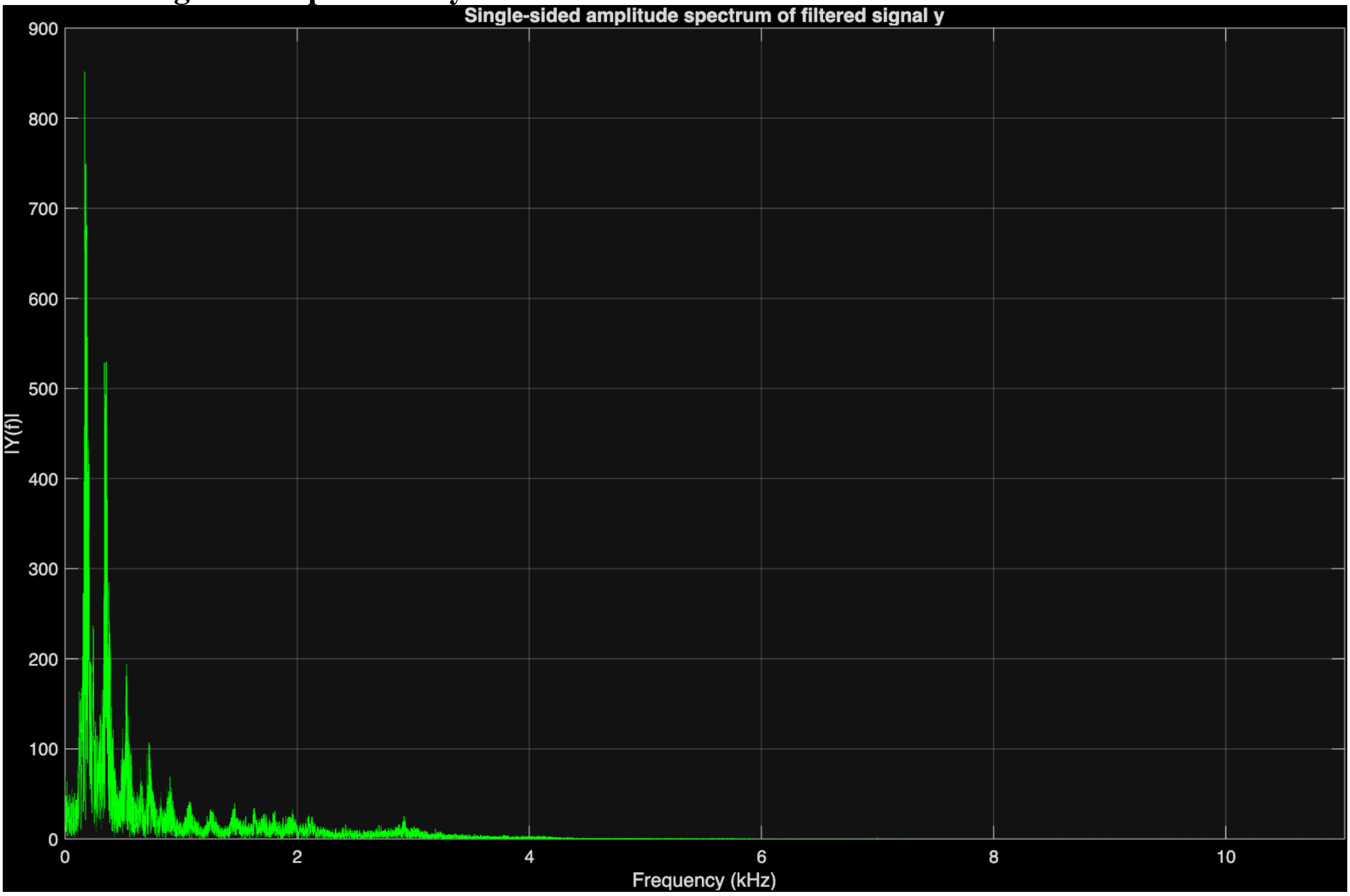
Sauvegarder en PDF et déposer sous Moodle	*Nom MARQUET* *Prenom Félix*

Expérience 1
Insérer le code ici

```
%% Experience 1 : Concatenate three audio files into one

clc;
clear;
close all;

%% Load audio data (Q2, holiday_offer, CLAVES)

[y1, Fs1] = audioread('Q2.wav');
[y2, Fs2] = audioread('holiday_offer.wav');
[y3, Fs3] = audioread('CLAVES.wav');

% Play sound (debug)
%sound(y1, Fs1); % Play the first audio file
%pause(length(y1)/Fs1); % Wait until the first audio finishes
%sound(y2, Fs2); % Play the second audio file
%pause(length(y2)/Fs2); % Wait until the second audio finishes
%sound(y3, Fs3); % Play the third audio file
%pause(length(y3)/Fs3); % Wait until the third audio finishes

%% Force all signals to mono and to the same sampling rate 44100 Hz (CD quality)

Fs_target = 44100;   % desired sampling frequency

% Convert to mono
if size(y1,2) > 1, y1 = mean(y1,2); end
if size(y2,2) > 1, y2 = mean(y2,2); end
if size(y3,2) > 1, y3 = mean(y3,2); end

% If needed, resample to 44100 Hz
if Fs1 ~= Fs_target
    y1 = resample(y1, Fs_target, Fs1);
end
if Fs2 ~= Fs_target
    y2 = resample(y2, Fs_target, Fs2);
end
if Fs3 ~= Fs_target
    y3 = resample(y3, Fs_target, Fs3);
end

%% Concatenate the three sounds

y_all = [y1; y2; y3];

%% Save combined sound

audiowrite('Q1marquetfelix.wav', y_all, Fs_target);

%% Play the combined sound

sound(y_all, Fs_target);
```

Expérience 2

Insérer la figure du spectre de y ici



Insérer le code ici

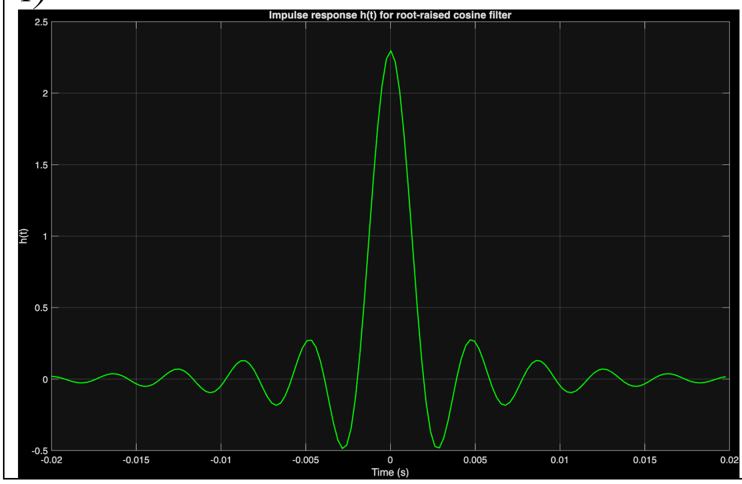
```
%% Experience 2 : Remove high–frequency jamming from Q2noiz810KA

clc;
clear;
close all;

%% 1) Load audio file with audioread

[x, fs] = audioread('Q2noiz810KA.wav');

% Convert to mono if stereo
if size(x,2) > 1
    x = mean(x,2);
end

N = length(x);

%% 2) Listen to the noisy signal

soundsc(x, fs); % Not working on my computer

%% 3) Visualize the spectrum of x and locate the jamming

X = fft(x);
magX = abs(X);

halfN = floor(N/2) + 1;
magX_half = magX(1:halfN);
f = (0:halfN–1) * (fs/N);    % frequency axis in Hz
```

```
figure('Name','Spectrum of x (noisy signal)');
plot(f/1000, magX_half, 'g');           % in kHz
xlabel('Frequency (kHz)');
ylabel('|X(f)|');
title('Single-sided amplitude spectrum of x');
grid on;
xlim([0 fs/2000]);

%% 4) Find the jamming frequency and filter

fjam = 4000;
ordre = 6;
[b, a] = butter(ordre, fjam/(fs/2), 'low');
y = filter(b, a, x);

%% 5) Listen to the filtered signal

soundsc(y, fs);                    % Not working on my computer
%audiowrite('test.wav', y, fs);  % To avoid using soundsc

%% 6) Plot spectrum of y

Y = fft(y);
magY = abs(Y);
magY_half = magY(1:halfN);

figure('Name','Spectrum of y (filtered signal)');
plot(f/1000, magY_half, 'g');         % in kHz
xlabel('Frequency (kHz)');
ylabel('|Y(f)|');
title('Single-sided amplitude spectrum of filtered signal y');
grid on;
xlim([0 fs/2000]);
```

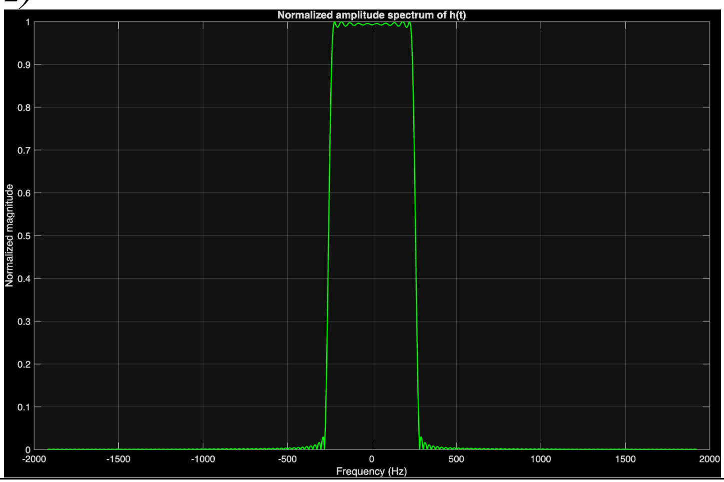
Expérience 3

Insérer les figures

1)



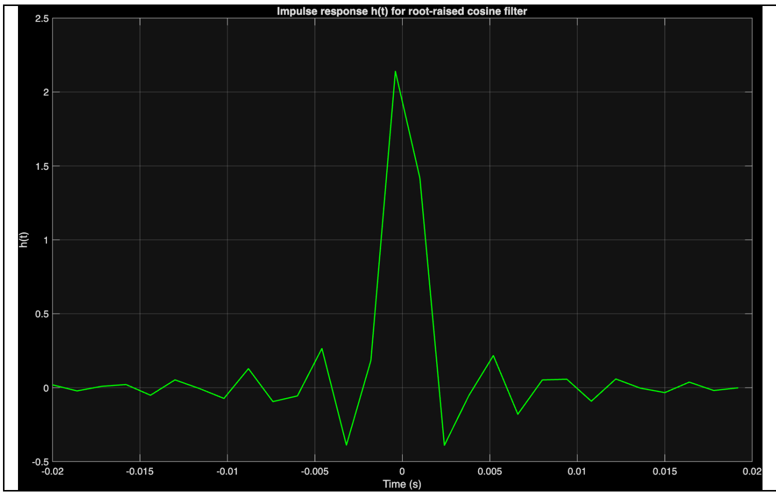
2)

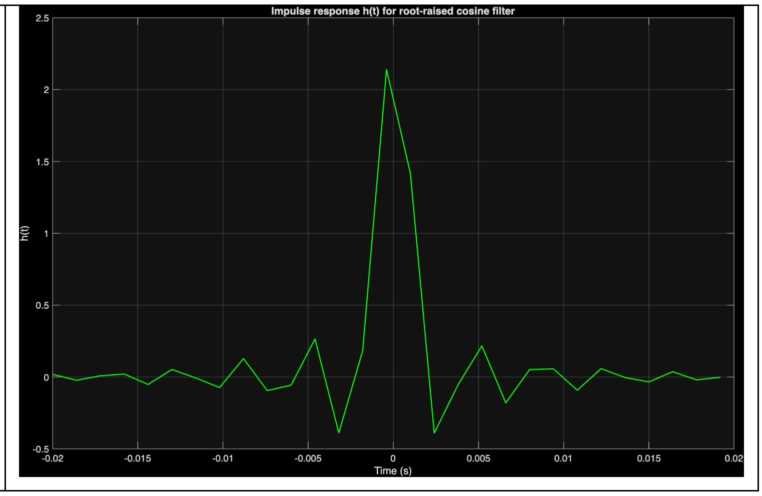


3) Quelle est la valeur maximale à ne pas dépasser ? Justifier
Fréquence max du filtre : *fmax = (1 + alpha) / (2T)*
Théorème de Nyquist : il faut *Fs >= 2*fmax* donc *Ts <= 1 / (2*fmax) = T / (1 + alpha)*
Avec *T = 2* ms et *alpha = 0.25* : *Ts_max = 2* ms */ 1.25 = 1.6* ms
Donc la valeur maximale à ne pas dépasser est 1.6 ms pour éviter l'aliasing.

4) **Insérer les figures ici**

h(t)	*2) Son spectre d'amplitude*





Insérer le code ici

```
%% Experiment 3 : Root-raised cosine (RRC) filter (subject 1)

clc;
clear;
close all;

%% Parameters

T   = 2e-3;              % Symbol period (2 ms)
al  = 0.1;               % Roll-off factor alpha (subject 1)
Te  = 1.4e-3;            % Sampling period (s) (near 1.6ms)
Fs  = 1 / Te;            % Sampling frequency (Hz)

t = -10*T : Te : 10*T;   % Time vector from -10T to +10T

%% Impulse response h(t) using the given formula

num = sin(pi*t/T * (1 - al)) + (4*al*t/T) .* cos(pi*t/T * (1 + al));
den = (pi*t/T) .* (1 - (4*al*t/T).^2);

h = (al / sqrt(T)) * (num ./ den);

% Handle singularity at t = 0 using theoretical limit
h(t == 0) = (1/sqrt(T)) * (1 - al + 4*al/pi);

%% 1) Time-domain visualization of h(t)

figure('Name','Impulse response h(t)');
plot(t, h, 'g', 'LineWidth', 1.2);
grid on;
xlabel('Time (s)');
ylabel('h(t)');
title('Impulse response h(t) for root-raised cosine filter');

%% 2) Amplitude spectrum of h(t)

NFFT = 2^12;                            % FFT length
H_freq = abs(fftshift(fft(h, NFFT))); % Magnitude spectrum (centered)
f = linspace(-Fs/2, Fs/2, NFFT);       % Frequency axis (Hz)

figure('Name','Amplitude spectrum of h(t)');
plot(f, H_freq / max(H_freq), 'g', 'LineWidth', 1.2);
grid on;
xlabel('Frequency (Hz)');
ylabel('Normalized magnitude');
title('Normalized amplitude spectrum of h(t)');

%% Theoretical maximum frequency of the RRC filter

f_max_theoretical = (1 + al) / (2 * T);    % Hz
```

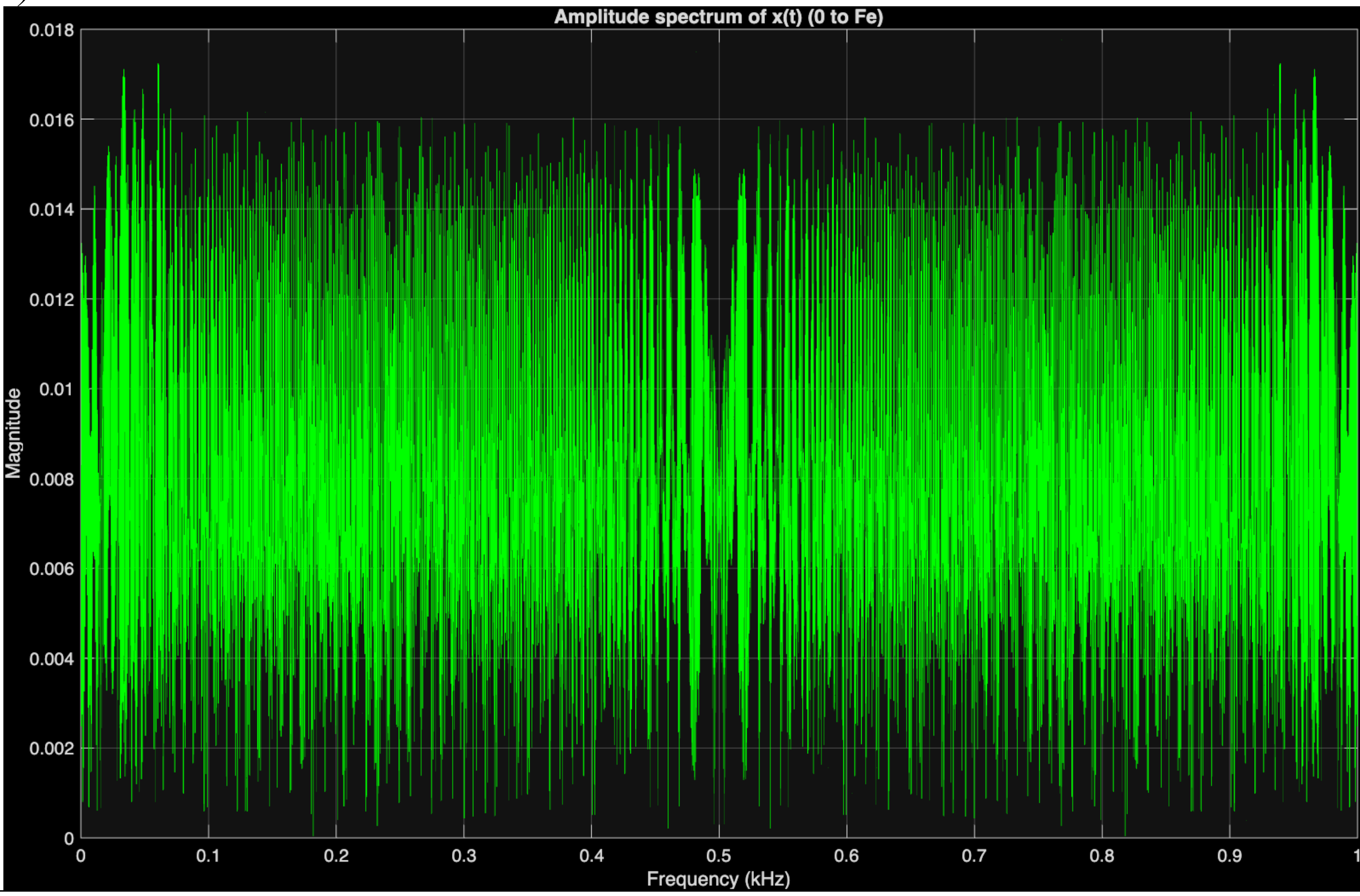
```
fprintf('Theoretical maximum frequency : %.2f Hz (%.3f kHz)\n', ...
        f_max_theoretical, f_max_theoretical/1000);
```

Expérience 4

1) Un tel échantillonnage n'est pas règlementaire. Expliquer pourquoi (1 point)

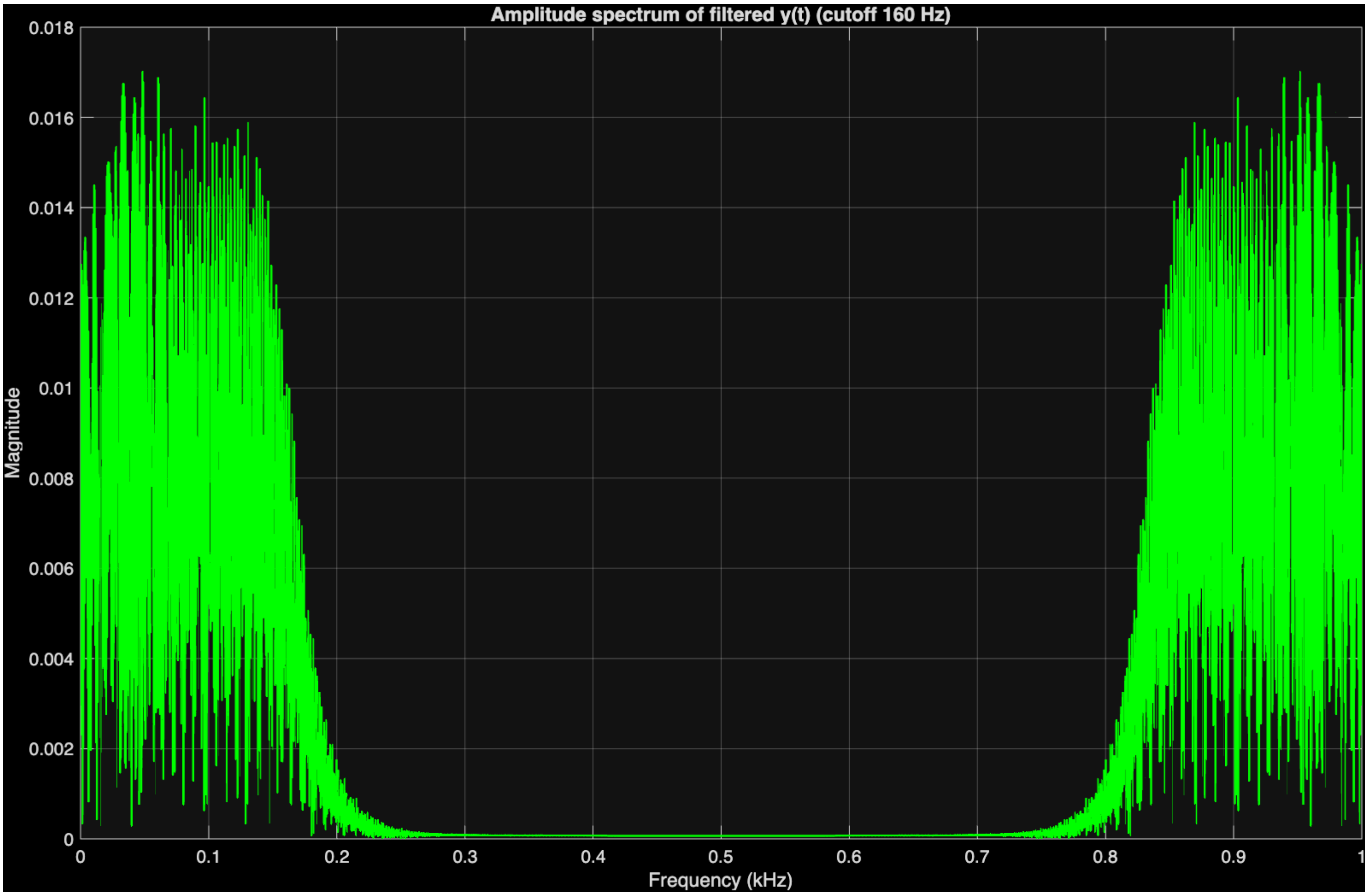
Le signal contient des composantes à *124000 rad/s*, soit des fréquences bien supérieures à $Fe/2 = 500\ Hz$. Les conditions du théorème de Nyquist ($Fe \geq 2*fmax$) ne sont donc pas respectées, ce qui provoque un repliement des composantes hautes fréquences dans la bande de base

Insérer les figures ici

2)	*3)* 

4)

Insérer la figure ici



Insérer le code ici

```
%% Experiment 4 : Sampling and filtering (subject 1)

clc;
clear;
close all;

%% 1) Sampling parameters

Fe   = 1000;                % Sampling frequency (Hz) for subject 1
Te   = 1 / Fe;              % Sampling period (s)
t    = (0:6400) * Te;       % Observation window (0 to 6400 samples -> 6.4 s)

%% 2) Generate x(t)
% x(t) = sin(2000 t^2) + sin(2000 t)^2/100 + sin(2000 t)/100
%         + cos(124000 t + pi/2) + sin(124000 t)

x = sin(2000*t.^2) ...
  + (sin(2000*t).^2)/100 ...
  + sin(2000*t)/100 ...
  + cos(124000*t + pi/2) ...
  + sin(124000*t);

%% 3) Amplitude spectrum of x(t) (0..Fe, axis in kHz)

N    = length(x);
Xfft = abs(fft(x)) / N;                 % linear magnitude spectrum
f_Hz = (0:N-1) * (Fe/N);                % frequency axis from 0 to Fe (Hz)
f_kHz = f_Hz / 1000;                    % in kHz

figure('Name','Amplitude spectrum of x(t)');
plot(f_kHz, Xfft, 'g');
title('Amplitude spectrum of x(t) (0 to Fe)');
xlabel('Frequency (kHz)');
ylabel('Magnitude');
xlim([0 Fe/1000]);                      % 0 .. Fe in kHz
grid on;
```

```matlab
%% 4) 8th-order Butterworth low-pass filter (cutoff adapted)

% In the original statement Fe = 5 kHz with Fc = 800 Hz.
% Here Fe is 5 times smaller, so we scale the cutoff: Fc = 800/5 = 160 Hz.
Fc    = 160;                          % cutoff frequency (Hz)
order = 8;                            % filter order
Wn    = Fc / (Fe/2);                  % normalized cutoff (0..1)

[b, a] = butter(order, Wn, 'low');    % Butterworth low-pass filter

%% 5) Filter the signal to obtain y(t)

y = filter(b, a, x);

%% 6) Amplitude spectrum of filtered signal y(t)

Yfft   = abs(fft(y)) / N;
figure('Name','Amplitude spectrum of filtered signal y(t)');
plot(f_kHz, Yfft, 'g', 'LineWidth', 1.2);
title(['Amplitude spectrum of filtered y(t) (cutoff ', num2str(Fc), ' Hz)']);
xlabel('Frequency (kHz)');
ylabel('Magnitude');
xlim([0 Fe/1000]);
grid on;
```