Signaux et Systèmes Électroniques

Signaux avec Matlab

Auteurs :
M. Samuel Riedo

Encadrant :
M. Daniel Oberson

19 octobre 2016

1 Signal entré point à point

Le premier exercice consiste à créer des vecteurs, puis les afficher à l'aide de plot ou stem.

```
%% Section 3.1

xaxis=[-4 -3 -2 -1 0 1 2 3 4 5 6];
yaxis=[0 0 0 0 4 3 2 1 0 0 0];

plot(xaxis, yaxis), grid on
hold on;                                % Without hold on, stem graph erase plot graph
stem(xaxis, yaxis), grid on

legend('plot','stem')                   % Graph comments
title(legend,'Exercice 3.1')
xlabel('x[ln]')
ylabel('n')
```

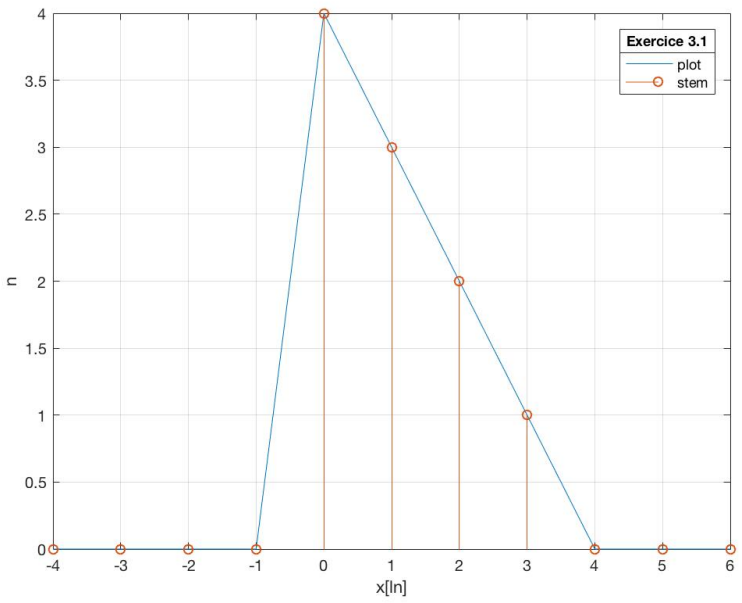

FIGURE 1 – Deux manières de plotter un signal.

La fonction plot relie les points par un trait continu, alors que stem non.

2 Série de Fourier

```
%% Section 3.2

t = (-1:10e-5:1);                          % -1 < t < 1 with 10us incr
f = 2;
y = 2*pi*f*t;
plot(t, (sin(2*pi*f*t)) + ((1/3)*sin(2*pi*3*f*t)) + ((1/5)*sin(2*pi*5*f*t)) + ((1/7)*sin(2*
    pi*7*f*t))), grid on
hold on;
plot(t, cos(y) + cos(3*y)/9 + cos(5*y)/25 + cos(7*y)/49), grid on

legend('x1','x2')                          % Graph comments
title(legend,'Exercice 3.2')
xlabel('Voltage')
ylabel('Time')
```

Les équations suivantes permettent de créer n'importe quel signal en sommant un certain nombre de sinus.

$$x_1 = \sum_{i=0}^{n-1} \frac{1}{2n+1} \cdot \sin(2\pi f t \cdot (2n+1)) \qquad x_2 = \sum_{i=0}^{n-1} \frac{1}{(2n+1)^2} \cdot \cos(2\pi f t \cdot (2n+1)^2)$$

La première équation x_1 permet d'obtenir des signaux carrés. Si n vaut 1, le signal sera un parfait sinus. Plus sa valeur augmente, plus il ressemblera à un signal carré (voir figure 2 avec $n = 4$).
La deuxième équation x_2 permet elle d'obtenir des signaux triangulaires en augmentant la valeur de n (voire figure 2 avec $n = 4$).

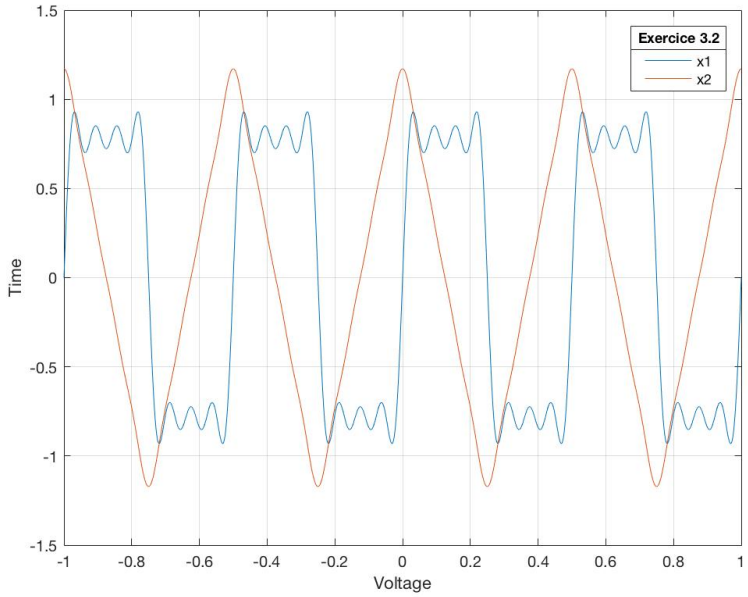

FIGURE 2 – Série de Fourier.

Il est néanmoins impossible d'avoir un signal parfait. Si la valeur de n est trop haute, le phénomène de Gibbs apparaît, se qui pique la valeur en y de la fonction lors d'un passage d'un zéro logique à un un logique.

3 Modulation

```
% Part 1

t = (0:5e-6:0.005);                         % 0 < t < 5 with 5us incr
y = 2*pi*440*t;
x1 = 0.6*cos(y)+0.3*cos(2*y + 1);
x2 = 0.1*cos(2*pi*25000*t);

x = (1+x1).*x2;
plot (t, x), grid on
legend('x')                                 % Graph comments
title(legend,'Exercice 3.3 Part 1')
xlabel('Time')
ylabel('Voltage')

% part 2

subplot(2, 1, 1); plot (t, x1), hold on, plot (t, x2)
legend('x1', 'x2')                          % Graph comments
title(legend,'Exercice 3.3 Part 2')
xlabel('Time')
ylabel('Voltage')

subplot(2, 1, 2); plot (t, x)
legend('x')                                 % Graph comments
title(legend,'Exercice 3.3 Part 2')
xlabel('Time')
ylabel('Voltage')
```

La modulation d'amplitude (Amplitude Modulation, AM) est une technique utilisée pour transmettre de l'information par onde radio, généralement du son. Le signal x_2 est une onde porteuse. Son amplitude va être modulée afin de transporter le signal x_1 (voir figure 3). Le signal modulé x est donné par la formule suivante :

$$x = (1 + x_1) \cdot x_2$$

L'intérêt d'utiliser un signal porteur avec une plus grande fréquence que le signal à émettre est d'augmenter la puissance d'émission, et donc de permettre à des récepteurs plus éloignés, ou avec des obstacles entre eux et l'émetteur, de capter le signal.

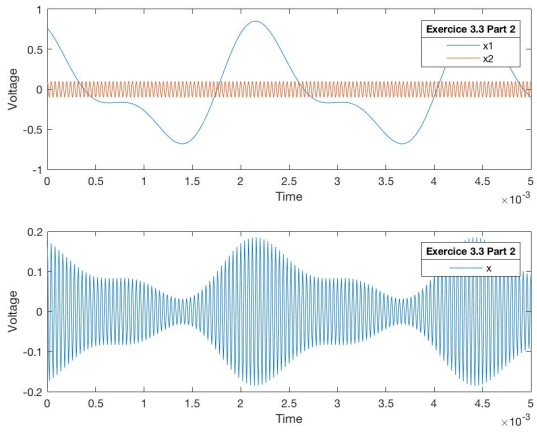

FIGURE 3 – Signal porteur.

4 Optionnel

```
% 3.4

% Part 1
t = (-1:10e-6:1);                          % -1 < t < 1 with 10us incr
plot(t, InverseFourier(250)), grid on

% Part 2
t = (0:5e-6:5);                            % 0 < t < 5 with 5us incr
y = 2*pi*440*t;
sound(0.6*cos(y)+0.3*cos(2*y + 1), 200000)  % 200000 because 1/5e-6 in t

% Part 3
t = (0:5e-6:5);                            % 0 < t < 5 with 5us incr
z = [t(1: round((1/5)*length(t))) ones(1,round((4/5)*length(t)))];
k = 0.6*cos(y)+0.3*cos(2*y + 1);

sound(0.6*cos(y)+0.3*cos(2*y + 1).*z, 200000)

subplot(2, 1, 1); plot (t, k)
legend('Son sans attaque')                 % Graph comments
title(legend,'Attaque du son')
xlabel('Time')
ylabel('Voltage')

subplot(2, 1, 2); plot (t, k.*z)
legend('Son avec attauque')                % Graph comments
title(legend,'Attaque du son')
xlabel('Time')
ylabel('Voltage')
```

```
function out = InverseFourier(n)
% Create a Fourier signal with n sinus
t = (-1:1e-5:1);                           % -1 < t < 1 with 10us incr
x = 2*pi*2*t;
y = sin(x);

    for i = 1:2:(n-1)
        y = y+(1/(i +2)) * sin(x * (i +2));
    end
    out = y;
end
```

La fonction InverseFourier permet de créer un signal sommant n sinus, de manière à créer des signaux carrés. InverseFourier(1) créera donc un parfait sinus, alors que InverseFourier(4) reproduira la même courbe que x_2 à l'exercice 3.2 (voir figure 2). Il est possible, au moyen de sound, de jouer le signal sur les haut-parleurs d'un ordinateur. Il est important d'utiliser le même échantillonnage (200 000 dans ce cas) pour avoir une représentation fidèle du son.

$$FS = \frac{1}{t} = \frac{1}{5 \times 10^{-6}} = 200\,000$$

Les musiciens utilisent souvent un procédé appelé “attaque” pour faire varier l’amplitude, et par conséquent le volume, d’un son à son commencement. Généralement, lorsqu’une note est pressée, le volume n’est pas directement à 100%, mais augmente dans le temps. La courbe d’attaque z ci-dessus faire varié l’intensité d’un signal entre 0 et 100% de manière linéaire sur une seconde.

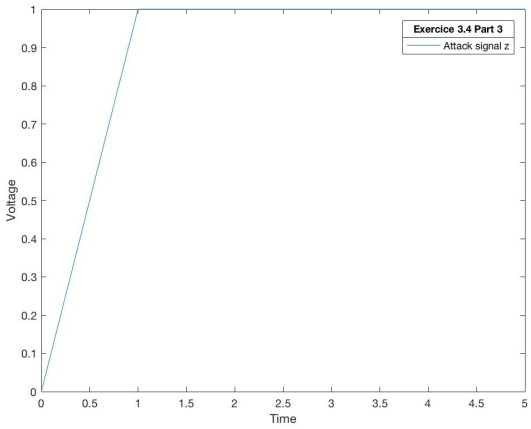

FIGURE 4 – Signal d’attaque du son.

Afin d’appliquer cette attaque au son, il suffit de multiplier un signal avec z :

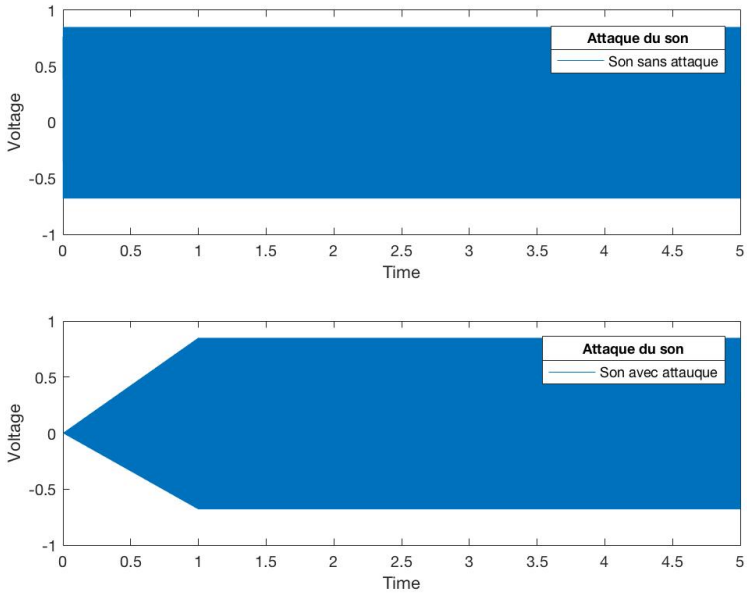

FIGURE 5 – Application de l’attaque sur le son