

TP N° 11 : Traitement de signal

Génération d'un signal

Nous allons voir les fonctions qui permettent de générer les signaux numériques usuels à l'aide de la fonction stem

1. Génération d'une impulsion unité

Afin de générer l'impulsion unité on peut écrire le script suivant et compiler le pour voir le résultat :

```
%impulsion unite
t=-10:10;
x=[zeros(1,10),1,zeros(1,10)];
stem(t,x);
axis([-10 10 -0.5 1.5]);
title('Impulsion unite');
xlabel('n');
ylabel('Amplitude');
```

2. L'échelon unité

Pour le cas de l'échelon unité, on se contentera d'un nombre fini d'échantillon. On peut écrire le programme suivant :

```
%echelon unite?e
t=-10:10;
x=[zeros(1,10),ones(1,11)];
stem(t,x);
axis([-10 10 -0.5 1.5]);
title('Echelon unite');
xlabel('n');
ylabel('Amplitude');
```

3. Autre exemple avec la fonction sinus

On recommence avec la fonction suivante $\sin(0,35 \times n)$. On écrira le programme suivant

```
%sinus
t=-10:10;
x=sin(0.35*t);
stem(t,x);
axis([-10 10 -1.5 1.5]);
title('sinus');
xlabel('n');
ylabel('Amplitude');
```

4. Autre exemple avec la fonction exponentielle

Ensuite avec une exponentielle décroissante $e^{0,2 \times n} \times u[n]$ avec u l'échelon unité. On a le programme suivant :

```
%exponentielle
t=-10:10;
u=[zeros(1,10),ones(1,11)];
x=exp(-0.2*t).*u;
stem(t,x);
axis([-10 10 -1.5 1.5]);
title('Exponentielle retard?e');
xlabel('n');
ylabel('Amplitude');
```

Opération sur les signaux

5. Décalage et retournement temporelle

Les signaux numériques sont souvent exprimés comme des combinaisons d'autres signaux élémentaires décalés ou retournés dans le temps. Ainsi le signal $s[n-N]$ est égal au signal $a[n]$ décalé de N échantillons dans le temps (N entier). On effectue donc une translation de N échantillons vers la droite, si N est positif, et vers la gauche si N est négatif. On va utiliser l'échelon unité pour illustrer ce concept. Le signal $a[-n]$ est la version retournée dans le temps de $a[n]$. Cela signifie que le signal est retourné par rapport au point $n = 0$.

On aura par exemple :

6. Décalage :

```
t=-10:10;
delta=zeros(1,10),ones(1,11);
subplot(3,1,1);
stem(t,delta);
axis([-10 10 -1.5 1.5]);
title('\delta[n]');
xlabel('n');
ylabel('Amplitude');
subplot(3,1,2);
deltam2=zeros(1,2),delta(1:end-2);
stem(t,deltam2);
axis([-10 10 -1.5 1.5]);
title('\delta[n-2]');
xlabel('n');
ylabel('Amplitude');
subplot(3,1,3);
deltap2=delta(3:end),zeros(1,2);
stem(t,deltap2);
axis([-10 10 -1.5 1.5]);
title('\delta[n+2]');
xlabel('n');
ylabel('Amplitude');
```

7. Retournement temporelle

```
t=-10:10;
u=zeros(1,10),ones(1,11);
x=exp(-0.2*t).*u;
subplot(2,1,1);
stem(t,x);
axis([-10 10 -1.5 1.5]);
title('x[n]');
xlabel('n');
ylabel('Amplitude');
subplot(2,1,2);
x1=x(end:-1:1);
stem(t,x1);
axis([-10 10 -1.5 1.5]);
title('x[-n]');
xlabel('n');
ylabel('Amplitude');
```

8. Addition, soustraction, multiplication et division sur les signaux

On peut maintenant aborder les opérations fondamentales sur les signaux. La fonction suivante permet d'ajouter deux signaux :

```
function [x,t]=sigadd(x1,t1,x2,t2)
%Adding two signals
%inputs :
% x1,t1 first input signal amplitudes and instants
% x2,t2 second input signal amplitudes and instants
```

```

%outputs :
% x,t sum signal amplitudes and instants
first=min(t1(1),t2(1));
last=max(t1(end),t2(end));
t=first:last;
%search number of preamble and postamble zeros
%needed to extend x1 and x2
preamb1=t1(1)-first;
preamb2=t2(1)-first;
postamb1=last-t1(end);
postamb2=last-t2(end);

%calcul des extensions de x1 et x2
if preamb1==0
    x1start=[];
else
    x1start=[zeros(1,preamb1)];
end

if preamb2==0
    x2start=[];
else
    x2start=[zeros(1,preamb2)];
end

if postamb1==0
    x1end=[];
else
    x1end=[zeros(1,postamb1)];
end

if postamb2==0
    x2end=[];
else
    x2end=[zeros(1,postamb2)];
end

%ajout des extensions
x1e=[x1start , x1 , x1end];
x2e=[x2start , x2 , x2end];

% faire la somme
x= x1e + x2e ;

```

La même chose pour les autres opérations

Activité 2 : Certain fonction Matlab

9. Lecture d'un signal

Lecture des exemples de données au taux d'échantillonnage différents

Exemple 1 :

```

load gong.mat;
sound(y);
sound(y,2*Fs)
sound(y,10*Fs)
sound(y,0.5*Fs)
sound(y,0.25*Fs)

```

Exemple 2 :

```

>>load handel.mat;
>> sound(y, Fs);
>>sound(y, 2*Fs);% à deux fois le taux d'échantillonnage enregistré.

```

```
>>sound(y, 2*Fs);% à Trois fois le taux d'échantillonnage enregistré.  
>>sound(y, 2*Fs);% à la moitié le taux d'échantillonnage enregistré.
```

Lecture de données d'échantillons avec une profondeur de bit spécifique

```
load handel.mat;  
nBits = 16;  
sound(y,Fs,nBits);
```

10. Transformation d'une matrice en un signal

```
>>y=1:10;  
>>sound(y, 2*Fs);
```

11. Fonctions playetaudioplayer

```
>> load handel;  
>> player = audioplayer(y, Fs);  
>> play(player);
```

12. Collectez un échantillon de votre discours avec un microphone et tracez les données du signal

```
% Record your voice for 5 seconds.  
recObj = audiorecorder;  
disp('Start speaking.')
```

```
recordblocking(recObj, 5);  
disp('End of Recording.');
```

```
% Play back the recording.  
play(recObj);
```

```
% Store data in double-precision array.  
myRecording = getaudiodata(recObj);
```

```
% Plot the waveform.  
plot(myRecording);
```

Autres Fonctions à voir

audiodevinfo	Information about audio device
audioplayer	Create object for playing audio
audiorecorder	Create object for recording audio
sound	Convert matrix of signal data to sound
soundsc	Scale data and play as sound
beep	Produce operating system beep sound

Références

http://www.morere.eu/IMG/pdf/tp_signal_gsi.pdf