BURNOT Jean-Christophe

COUTAUD Hugo

PEROL Julien

PEY Arthur

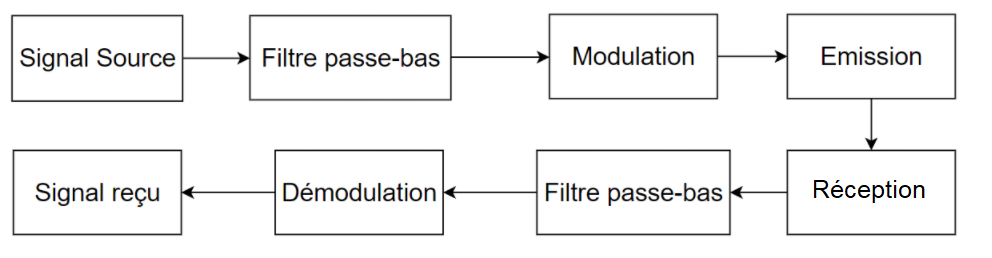
Compte rendu projet SSL

Modulation

**Sommaire :**

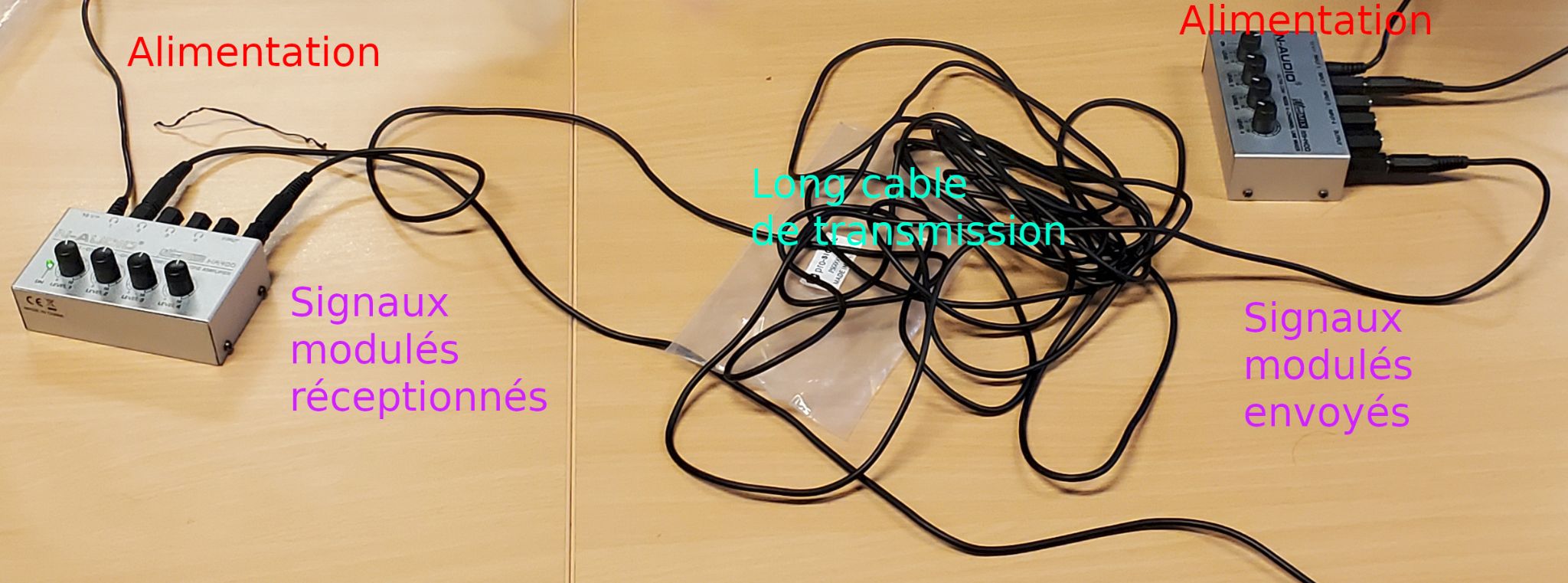
1. Bureau d’études : analyses et cahier des charges. 1
2. Modulation/démodulation d’un signal. 3
3. Modulation sans porteuse du signal 3
4. Démodulation sans porteuse du signal 5
5. Méthode BLU 6
6. Envoi de deux messages en simultané (Modulation BLU) 6
7. Modulation par la méthode BLU en pratique 7
8. Démodulation par la méthode BLU en pratique 8
9. Utilisation d’un multiplexeur et d’un démultiplexeur 10
10. Interprétation et analyse critique des résultats 10
11. Conclusion 11
12. Annexes 11
13. **Cahier des charges**

Le but du TP est de transmettre simultanément et sur un même canal 4 messages vocaux émis par 4 sources différentes et récupérés par 4 récepteurs différents. On nous restreint de plus sur une bande passante de Fe = 96 kHz. Le problème doit, entre autres, être résolu à l’aide d’une Modulation d’Amplitude sans Porteuse. La programmation se fait sous MatLab, plusieurs fonctions nous sont fournies .



*Schéma bloc du système complet.*

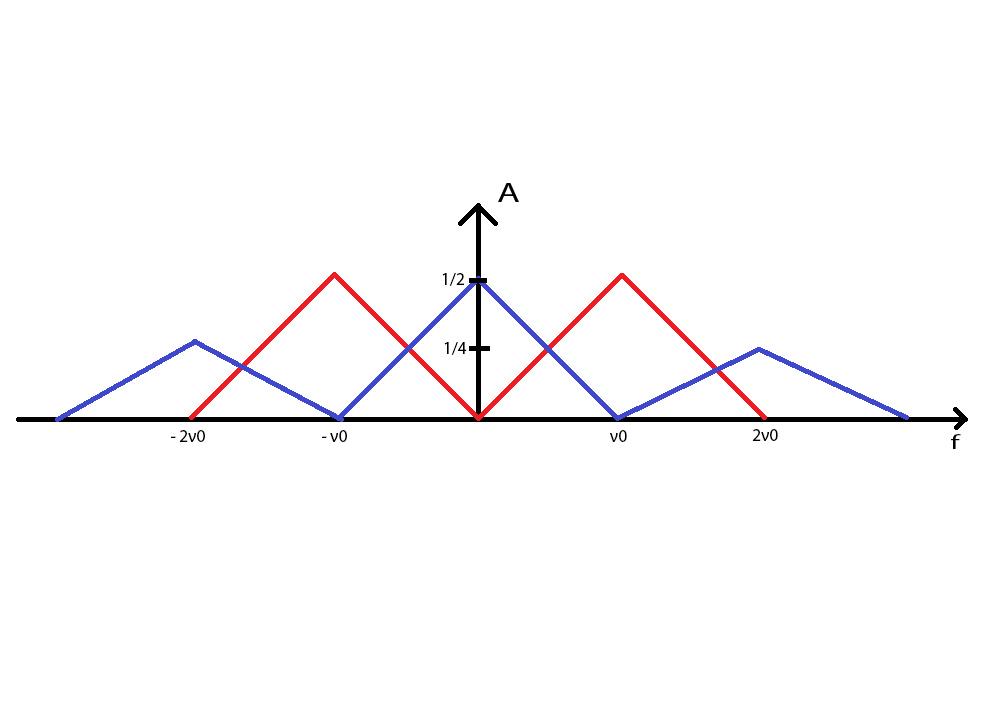
Dans un second temps, il faudrait transmettre 6 signaux simultanément sur la même bande passante, le tout en limitant au mieux toute perte de qualité sur les messages originaux. Tous les codes utilisés sont présents dans le zip contenant ce rapport.



Bureau d'Étude; création et affinement de la solution au problème.

Notre étude commence tout d’abord par la répartition de bandes fréquentielles entre les groupes. Durant les séances, nous avions travaillé à trois groupes sur le problème, mais notre étude sera présentée dans le cas de quatre groupes afin de rester fidèle à l’esprit de l’exercice.  
Il s’agit donc de garder en tête le besoin de se répartir des espaces de fréquence bien définis afin de ne pas empiéter les uns sur les autres lors de nos transmissions.

Nous commençons par nous intéresser à notre restriction principale, la bande passante maximale de Fe = 96 kHz. D’après le théorème de l’échantillonnage, il nous faut absolument respecter le critère de Shannon s’il l’on souhaite numériser un signal vocal sans perte majeures. Cela correspond à se concentrer sur une fréquence d’échantillonnage, notée Fe, telle que Fe > 2B.



Autrement dit, nos signaux devront être compris entre 0 et 48 kHz en valeur de B.

Ayant cette information en tête, il nous faut maintenant comprendre comment la Modulation d’amplitude sans porteuse fonctionne et ce qu’elle implique.

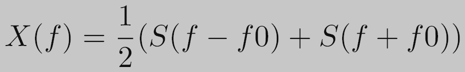
On note s(t) notre signal sonore d’origine. Il correspond au message vocal à transmettre.

En multipliant ce signal par la porteuse p(t) on module notre signal. On note x(t) le signal modulé.



Avec v0 la fréquence de la porteuse.

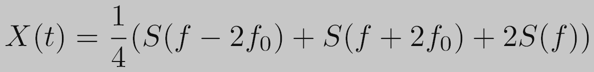
On a choisi de passer par la transformée de Fourier afin de mieux visualiser les signaux potentiels que l’on va envoyer ou reçevoir. Par transformation de Fourier de x(t), noté X(f), tombe sur le résultat suivant (signal rouge):



On remarque que ce signal est symétrique autour de nu0 et est compris sur l’intervalle [-2mu0; 2mu0], qui correspond à [-2B; 2B]. Si le signal reçu est présent sur un tel intervalle, on risque alors de sortir du critère de Shannon. Il faut prendre cela en compte dans nos valeurs, on prend ainsi B = 24 kHz.

On transmet ainsi ce signal et on reçoit Smod, le signal modulé.

La démodulation prend place lors de la multiplication de ce signal Smod avec la porteuse. On se retrouve alors avec le signal bleu dont l’équation est:



Le signal qui nous intéresse est entièrement présent sur un intervalle [-𝜈0; 𝜈0]. Dans la pratique, il nous faudrait revenir dans l’espace temporel, mais pour garder la simplicité de compréhension de notre schéma, nous restons en Fourier pour l’explication suivante.

D’après notre schéma, on récupère ensuite le signal bleu en lui appliquant un passe-bas filtrant tout signal hors de l’intervalle [-𝜈0; 𝜈0]. Il nous reste alors le signal d’origine en fréquentiel.

En pratique, il nous faut donc procéder à une transformation de Fourier Inverse sur le signal bleu, puis filtrer par un passe-bas.

Finalement, on doit se répartir un espace fréquentiel de longueur B = 24 kHz en quatre. Chaque groupe à donc à sa disposition une bande de 6 kHz centrées en 3 kHz pour le premier; 9 kHz pour le deuxième; 15 kHz pour le troisième et 21 kHz pour le quatrième.

Dans la pratique, à trois groupes, chacun possède une bande de largeur 8 kHz. On se centre en 4 kHz, 12 kHz ou 20 kHz selon le groupe.

Notre groupe a dû se centrer autour de la fréquence 4 kHz.

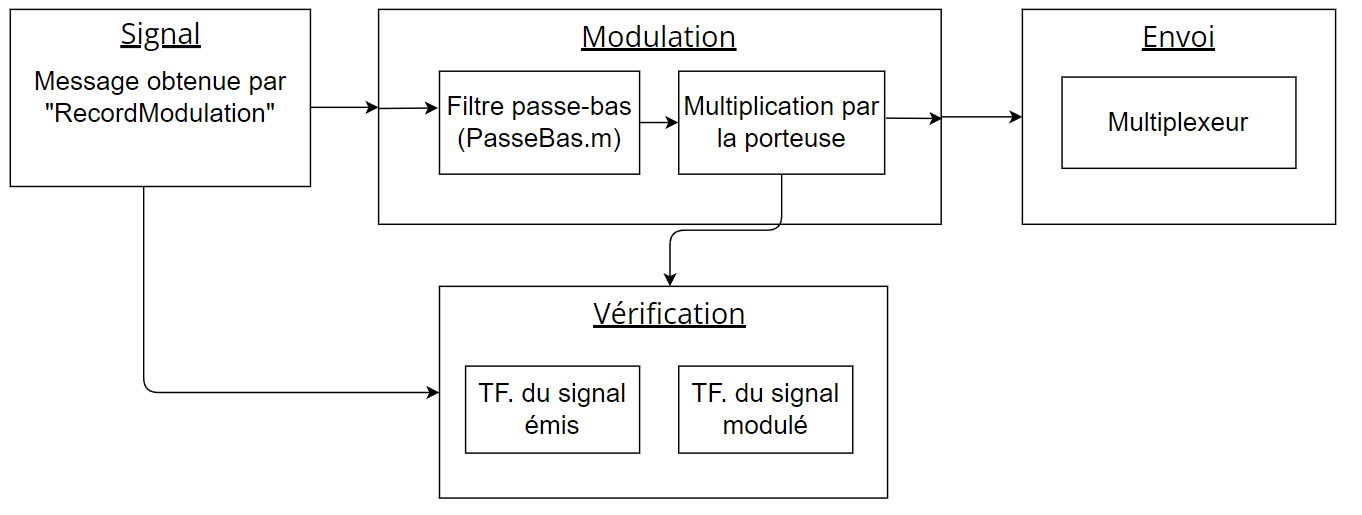
On tombe ainsi sur les paramètres suivants pour notre système :

* nu0 = 4000 hz (fréquence de la porteuse)
* Fe = 96000 hz (fréquence d'échantillonnage)
* Fc = 4000 hz (fréquence de coupure)

temps du message (arbitraire)= 5s

Longueur de la Bande allouée à notre groupe = 8khz

1. **Modulation/Démodulation sans porteuse**
2. **Modulation sans porteuse du signal :**

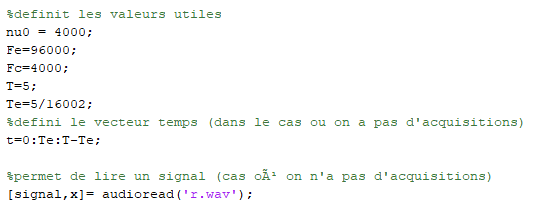


L’objectif est de moduler un signal préalablement enregistrer et filtrer pour obtenir un signal forcément compris dans notre bande fréquentiel attribuée [0Hz,8kHz].

Pour ce faire nous avons créé un code Matlab contenue dans le fichier *Emission.m*.

Nous enregistrons notre message à l'aide de la fonction *RecordModulation.m* fournie. Celle-ci permet de retourner deux liste:

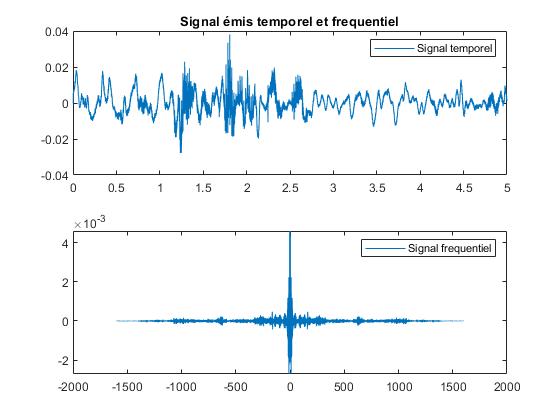
* Une liste de temps.
* L'amplitude du signal à tous les instants définie dans la liste de temps.



Après enregistrement, on obtient le signal temporel (signal qui sera envoyé).

On peut aussi afficher sa transformée de Fourier pour vérifier que le support

fréquentiel est effectivement compris dans l’intervalle [0Hz,8kHz].



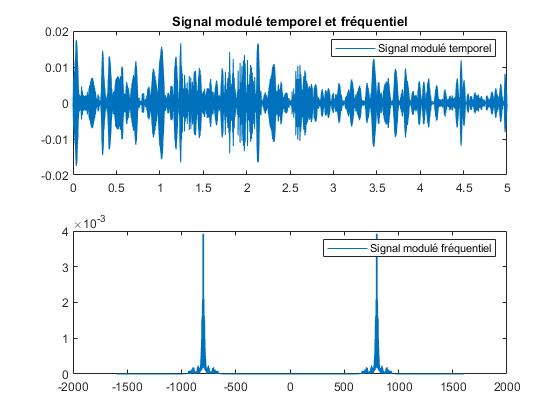
On constate que la fréquence maximum atteinte est d’environ 1,5kHz,

Dans notre cas, la fréquence maximale est donc inférieure à 4kHz. Cependant, pour éviter tous risques, un filtre passe-bas est ensuite appliqué pour supprimer les fréquences supérieures à 4 kHz qui pourraient interférer avec les autres messages envoyés sur le canal.

Enfin, nous pouvons moduler le signal en multipliant le signal par la porteuse : , avec

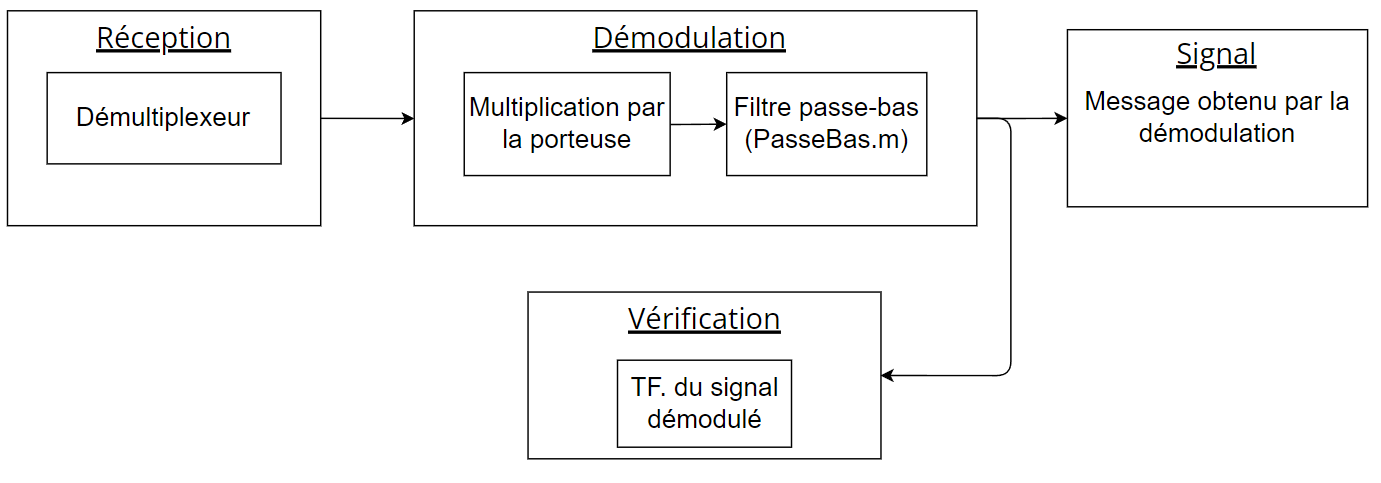


A l’aide d’une transformation de Fourier du signal, on peut en afficher son spectre et vérifier le support spectrale du signal.



Comme attendu, le signal modulé a une fréquence inférieure à 4kHz (et donc la totalité du signal s'étend sur une fréquence comprise dans l’intervalle intérieur a 8kHz ).

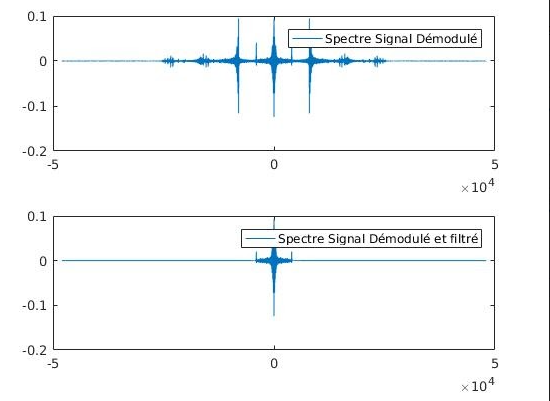
1. **Démodulation sans porteuse du signal:**



La partie démodulation du signal reçu comporte en réalité deux étapes; la démodulation et le filtrage, comme vu dans notre bureau d’étude.

Après démodulation par multiplication avec la porteuse, on utilise la fonction *PasseBas* fournie sur la fréquence de coupure Fc.

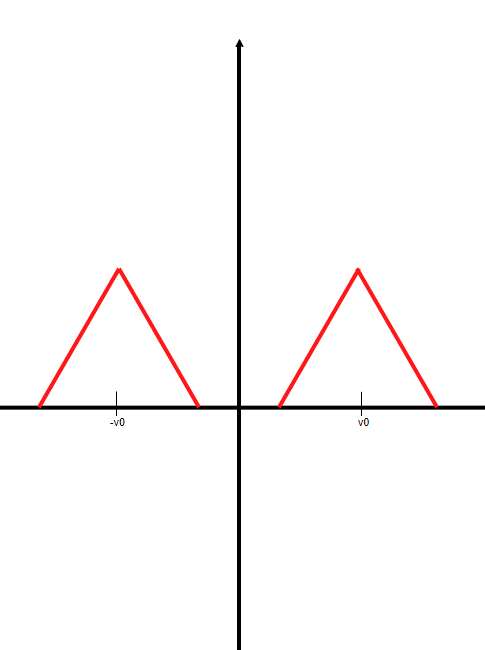
On obtient ainsi les graphiques suivants:



**3. Modulation BLU**

1. **Modulation de deux messages en simultané (Modulation BLU)**

Le but de cette partie est de présenter comment nous nous sommes organisés pour, à l’aide de changements mineurs à notre code d’origine, envoyer deux messages vocaux en simultanés. Pour cela, on choisit de procéder par Modulation à Bande Latérale Unique (BLU).

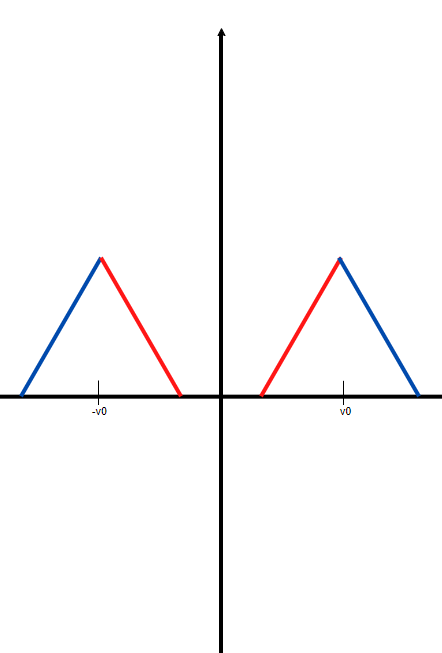
Ci-dessous un schéma qui représente le signal reçu après modulation. Par application d’un passe-bas sur [-𝝁0; 𝝁0] après démodulation, on récupère exactement les informations requises pour reconstituer le message original.

Autrement dit, on récupère la partie gauche et droite du triangle rouge.

Dans ce cas, il faut démoduler avant d'appliquer notre passe-bas.

Cependant, une telle méthode peut encore être optimisée. En effet, par démodulation, les parties extérieure à l’intervalle [-𝝁0; 𝝁0] sont inutiles, car en “double”.

Le but de la BLU se veut de profiter de ces données inutiles, en les remplaçant par des informations sur un second signal. Avec le bon filtre, on peut alors les récupérer et reconstituer ce second signal à démoduler en parallèle du premier.



Dans ce cas, il nous faut appliquer un passe-bande avant la démodulation, afin de conserver soit les informations du message 1, soit les informations du message 2.

Cette méthode nous permet ainsi de conserver une qualité de message similaire à la première partie tout en multipliant par deux le nombre de messages envoyés par chaque groupe. Dans la pratique, il nous faut modifier notre code pour pouvoir enregistrer deux messages, les moduler sur le même espace fréquentiel, puis démoduler et filtrer afin de pouvoir sélectionner le message voulu.

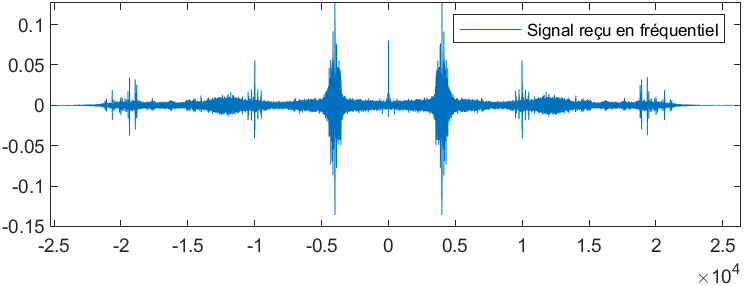
Toutes les valeurs utilisées sont précisées dans la conclusion du bureau d’études.

1. **Modulation par la méthode BLU en pratique:**

On commence l’opération par appliqué un passe-bas à nos deux messages afin de s’assurer de rester dans nos plages de fréquences restreintes plus tard. Vient ensuite la création des signaux modulés. Pour cela, on change légèrement la formule de modulation. Pour Signal1 notre signal filtré, on somme le terme {imag(hilbert(Signal1)) .\* sin(2\*pi\*nu0\*t))./max(abs(Signal1)} à notre modulation habituelle. Cela permet d’obtenir les fragments de signaux décrits plus haut.

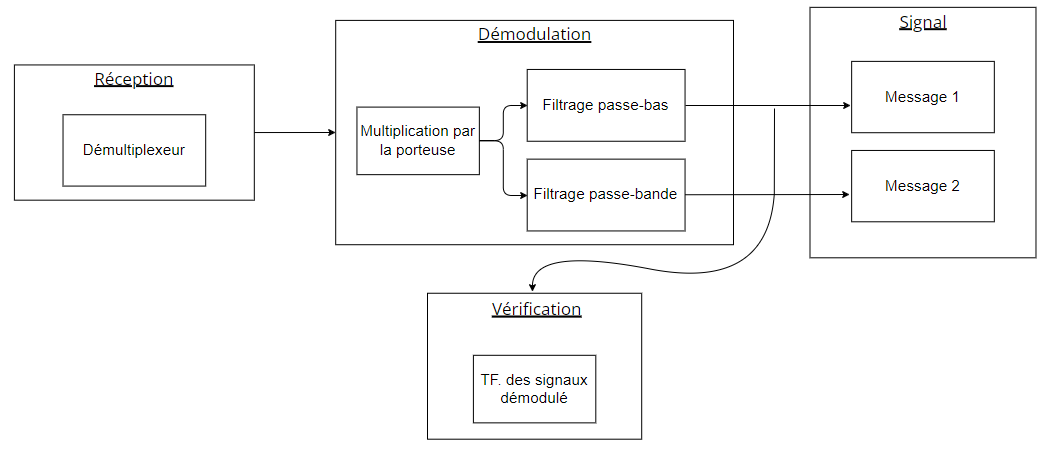
Pour notre Signal2, on soustrait cette fois le terme :

{imag(hilbert(Signal2)).\*sin(2\*pi\*nu0\*t))./max(abs(Signal2)} à notre modulation habituelle. Pour finir, on additionne les deux signaux modulés obtenus, puis on peut envoyer le tout.



On observe bien une dissymétrie entre les deux côtés d’un pique, marque de la présence d’information de deux messages.

1. **Démodulation par la méthode BLU en pratique:**

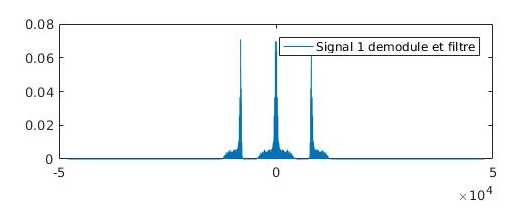


On arrive finalement à la démodulation.

Dans notre cas, il s’agit de récupérer le signal envoyé et d’en faire une copie.

On les nommes *SignalReçu* et *SignalReçuBis*.

On cherche en premier lieu à reconstituer le 1er message. On envoie donc ce signal *SignalReçu* dans un passe bande d’intervalle [𝝁0; 𝝁0 + LargueurAllouée/2]. On récupère ainsi les parties concernant le 1er message uniquement. On procède alors à la démodulation en multipliant par la porteuse, et on obtient notre premier message en fréquentiel. On peut l’entendre par la fonction *soundsc* de MatLab.

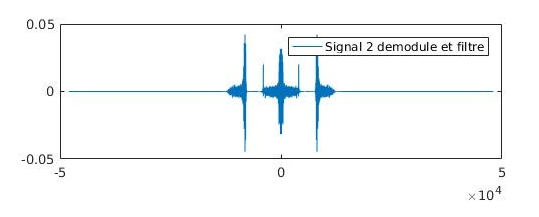


Il nous faut maintenant pouvoir reconstituer le deuxième message. Pour cela nous devons en théorie procéder comme avant mais avec un passe-bande d’intervalle [𝝁0-LargueurAllouée/2; 𝝁0]. Par construction de la fonction *PasseBande* fournie cependant, cela n’est pas possible (erreur interne lors d’un calcul).

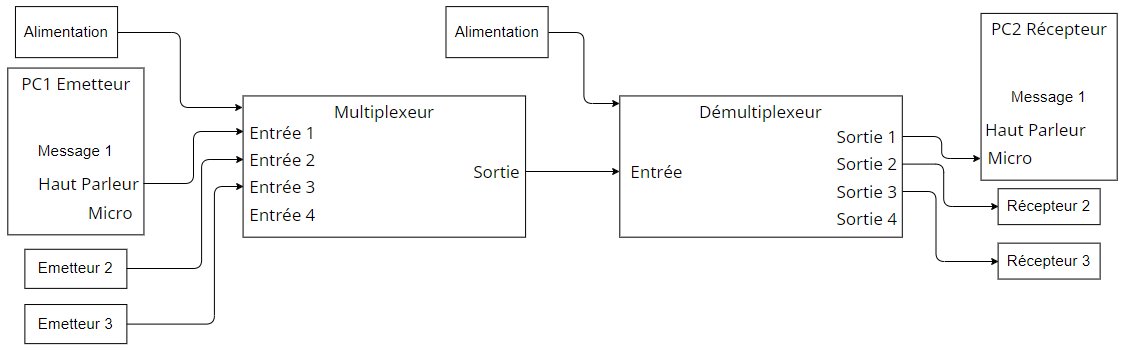
Pour obtenir ce deuxième message, on va passer par une solution alternative. On commence par démoduler immédiatement *SignalReçuBis*, puis on envoie le résultat dans un passe-bas de fréquence de coupure Fc. En procédant ainsi, on obtient un signal qui, lorsqu’il est joué, correspond à au message 1 superposé au message 2.

Il nous reste alors à retirer le message 1 de ce signal pour obtenir le message 2. On décide de lui soustraire le message 1 tel qu’on l’a obtenu en premier lieu.

Finalement, le signal ainsi obtenu correspond bien à notre message 2 originalement envoyé.



**4. Utilisation d’un multiplexeur et d’un démultiplexeur**



Dans la pratique, il nous a fallu utiliser un multiplexeur et un démultiplexeur pour centraliser et transmettre les messages de chaque groupe via la même interface.

Pour cela, il nous fallait brancher le port Casque de l’ordinateur en entrée du multiplexeur, et le port Micro en sortie du démultiplexeur. Il nous est alors possible de gérer l’amplitude sur le multiplexeur pour obtenir un message d’une qualité plus ou moins prononcé en fin d’opérations. Lors de nos tests, on a également remarqué que le passage par ces intermédiaires apportait des parasites sur nos messages reconstitués. Cela était probablement dû à un défaut du côté du matériel plutôt qu’à la technique.

**5. Interprétations et analyses critiques des résultats.**

La plupart de nos interprétations ont été présentées directement à la suite de nos explications pour permettre une mise en question de nos résultats de manière immédiate lorsque cela était nécessaire. Dans une optique de synthèse, nous n’allons pas dupliquer tout cela dans cette partie.

Cependant, les nombreux graphiques obtenus nous apportent une cohérence entre les résultats théoriques du bureau d’étude et les résultats pratiques amenés par nos codes MatLab. Nos signaux possèdent une symétrie visible lors de nos modulations et se comportent correctement lorsque l’on prend le temps de les observer à travers des transformées de Fourier diverses. L’impact de chacune de nos étapes est alors clairement visible sur ces derniers.

Une critique majeure sur notre projet repose sur la qualité approximative des signaux audios de sorties, ainsi que le manque de précision de nos graphiques (pas assez zoomés, manque d’annotations directes…). Là où le problème de l’audio peut se justifier par un problème de matériel capricieux, les imprécisions de nos graphiques sont surtout liées à notre incapacité à les reproduire dans des conditions plus contrôlées.

En effet, certaines opérations nécessitent l’utilisation de multiplexeurs ou autres, qui nous sont fournis par l’école lors des TPs. Ayant cela en tête, nous avons, là où cela était possible, repris nos codes Matlab et relancé nos exécutions pour obtenir des résultats plus lisibles.

**6. Conclusion**

Grâce à ce TP où nous avons expérimenté la démodulation sans porteuses, nous avons pu dans un premier temps transmettre un signal sonore entre 2 machines. Puis, en ayant utilisé une fréquence de porteuse adéquate, nous avons pu transmettre 3 signaux simultanément à l’aide d’un multiplexeur.

Enfin, nous avons pu constater l’efficacité de Bande Latérale Unique qui permet de transmettre le double de signaux pour une même Bande passante et une même qualité de réception.

**7. Annexes**

-Documents de cours disponibles sur le E-Campus

-Fichier Matlab: *RecordModulation.m PasseBande.m, PasseBas.m, TransFourier.m, TransFourierInv.m*

-Référence du multiplexeur: N-AUDIO MX-400

-Référence du Démultiplexeur: N-AUDIO HA-400

Code modulation :

% Portion de code qui permet de générer le signal.

%definit les valeurs utiles

nu0 = 4000;

Fe=96000;

Fc=4000;

T=5;

Te=5/16002;

%defini le vecteur temps (dans le cas ou on a pas d'acquisitions)

t=0:Te:T-Te;

%permet de lire un signal (cas oÃ¹ on n'a pas d'acquisitions)

[signal,x]= audioread('r.wav');

%Permet d'acquÃ©rir le signal

%[nomfic,Signal,t] = RecordModulation(Fe,T);

%[nomfic2,Signal2,t2] = RecordModulation(Fe,T);

sfiltr=PasseBas(signal,Fe,Fc); % filtrage du signal 1

porteuse = cos(2\*pi\*nu0\*t);

smod = sfiltr .\* porteuse' ./ max(abs(sfiltr)); % modulation du signal filtre

%Permet d'envoyer le signal en boucle

%while 0==0;

%soundsc(smod, Fe);

%pause(T);

%end

[S,f] = TransFourier(signal,t');

[Sfiltr,ffiltr] = TransFourier(sfiltr,t');

[Smod,fmod] = TransFourier(smod, t');

figure(1);

subplot(211);

plot(t,signal);

title("Signal émis temporel et fréquentiel")

legend("Signal temporel")

subplot(212);

plot(f,S);

legend("Signal fréquentiel")

figure(2); hold on;

subplot(211);

plot(t,sfiltr);

title("Signal émis filtré temporel et fréquentiel")

legend("Signal filtré temporel")

subplot(212);

plot(ffiltr,Sfiltr');

legend("Signal filtré fréquentiel")

figure(3);

subplot(211);

plot(t,smod);

title("Signal modulé temporel et fréquentiel");

legend("Signal modulé temporel")

subplot(212);

plot(fmod,Smod);

legend("Signal modulé fréquentiel")

Code démodulation :

%%

%définit les valeurs utiles

nu0 = 4000;

Fe=96000;

Fc= 4000;

BP = 8000;

temps = 5;

[nomfic,Signal,t] = RecordModulation(Fe,temps);

%définit la porteuse

porteuse = cos(2\*pi\*nu0\*t);

%%

%réception du message, démodulation

signalDemodule = porteuse .\* Signal;

%Obtention du signal en sortie filtre

[signalDemoduleFiltre]=PasseBas(signalDemodule,Fe,Fc);

%Ecoute du message

%while 0 == 0

%soundsc(signalDemoduleFiltre,Fe);

%pause(temps);

%end

%%

%AFFICHAGES

%Set up du vecteur temps (dans le cas ou on ne rÃ©cupÃ¨re pas le signal)

%[audio,Fe]=audioread('r.wav');

%audio = audio(:,1)'; %Pour eviter les enregistrements stï¿½rï¿½o

%T=length(audio)\*1/Fe;

%t=0:1/Fe:T-1/Fe;

figure(1);

subplot(211);

plot(t,Signal);

legend('Signal reÃ§u en temporel');

figure(2);

[spectre1,f] = TransFourier(Signal,t);

subplot(211);

plot(f,abs(spectre1));

legend('Spectre Signal ModulÃ©');

figure(3);

[spectre2,f]=TransFourier(signalDemodule, t);

subplot(211);

plot(f,spectre2);

legend('Spectre Signal DÃ©modulÃ©');

[spectre3,f]=TransFourier(signalDemoduleFiltre, t);

subplot(212);

plot(f,spectre3);

legend('Spectre Signal DÃ©modulÃ© et filtrÃ©');

Code modulation(BLU) :

%%

%\_\_\_\_\_\_\_\_\_\_

%Portion de code qui permet de gÃ©nÃ©rer le signal.

%dÃ©finit les valeurs utiles

nu0 = 4000;

Fe=96000;

Fc=4000;

T=5;

%Permet de lire un fichier 'x' dÃ©jÃ enregistrÃ©

%[x,Fe]= audioread(â€˜nomfichier.wavâ€™)

%Permet de faire les enregistrements

[nomfic,Signal,t] = RecordModulation(Fe,T);

[nomfic2,Signal2,t2] = RecordModulation(Fe,T);

%definit la porteuse

porteuse = cos(2\*pi\*nu0\*t);

%Obtention des signaux en sortie de filtre

sfiltr=PasseBas(Signal,Fe,Fc); % filtrage du signal 1

sfiltr2=PasseBas(Signal2,Fe,Fc); % filtrage du signal 2

%GÃ¨re les signaux de la BLU

smod=(sfiltr .\* porteuse - imag(hilbert(sfiltr)) .\* sin(2\*pi\*nu0\*t))./max(abs(sfiltr));

smod2=(sfiltr2 .\* porteuse + imag(hilbert(sfiltr2)) .\* sin(2\*pi\*nu0\*t))./max(abs(sfiltr2));

smodTot = smod + smod2;

%Permet d'envoyer en boucle le signal

while 0==0;

soundsc(smodTot, Fe);

pause(T);

end;

%%

%\_\_\_\_\_\_\_\_\_\_

%Partie qui gÃ¨re la vÃ©rification des signaux

[S,f] = TransFourier(Signal,t);

[Sfiltr,ffiltr] = TransFourier(sfiltr,t);

[Smod,fmod] = TransFourier(smod, t);

[SmodTot,fmodTot] = TransFourier(smodTot, t);

%AFFICHAGES

figure(1);

subplot(211);

plot(t,Signal);

subplot(212);

plot(f,S);

figure(2);

subplot(211);

plot(t,sfiltr);

subplot(212);

plot(ffiltr,Sfiltr);

figure(3);

subplot(211);

plot(t,smod);

subplot(212);

plot(fmod,Smod);

figure(4);

plot(fmodTot,SmodTot);

Code démodulation(BLU) :

%%

%\_\_\_\_\_\_\_\_\_\_

%Portion de code qui permet de rÃ©cupÃ©rer le signal.

%définit les valeurs utiles

nu0 = 4000;

Fe=96000;

Fc=4000;

temps = 5;

BP = 8000;

%Permet de lire un fichier 'x' dÃ©jÃ enregistrÃ©

%[x,Fe]= audioread(â€˜nomfichier.wavâ€™)

%Permet d'enregistrer le signal reÃ§u

[nomfic,Signal,t] = RecordModulation(Fe,temps);

%définit la porteuse

porteuse = cos(2\*pi\*nu0\*t);

%Pour Ã©couter le signal filtrÃ©

%soundsc(smod, Fe);

%%

%\_\_\_\_\_\_\_\_\_\_

%reception du message, demodulation

%Permet de demoduler le signal (passe bas)

signaltest = porteuse .\* Signal;

%Obtention des signaux en sortie de filtre

[signalDemoduleFiltre]=PasseBande(Signal,Fe, nu0+0.0001 - BP/2,nu0); %Donne rien

[signalDemoduleFiltre2]= PasseBande(Signal,Fe,nu0, nu0+BP/2); %Donne le 1er message

[SignalTest]=PasseBas(signaltest,Fe,Fc); %Donne le 1er et le 2Ã¨me

%Demodule les signaux (passe bande)

signalDemodule = porteuse .\* signalDemoduleFiltre;

signalDemodule2 = porteuse .\* signalDemoduleFiltre2;

%Fais la diffÃ©rence des signaux pour obtenir le 2Ã¨me signal

signalSomme = SignalTest - signalDemodule2;

%Permet de de lire en bloucle les diffÃ©rents signaux

%while 0 == 0

%soundsc(real(signalDemodule),Fe);

%pause(temps);

%soundsc(real(signalDemodule2),Fe);

%pause(temps);

%soundsc(real(SignalTest),Fe);

%pause(temps);

%soundsc(real(signalSomme),Fe);

%pause(temps);

%soundsc(real(SignalTest2),Fe);

%pause(temps);

%end

%%

%\_\_\_\_\_\_\_\_\_\_

%AFFICHAGES

%Set up du vecteur temps dans le cas oÃ¹ on n'y a pas accÃ¨s.

%[audio,Fe]=audioread('r.wav');

%audio = audio(:,1)'; %Pour eviter les enregistrements stereo

%T=length(audio)\*1/Fe;

%t=0:1/Fe:T-1/Fe;

%fait les diffÃ©rents affichages:

%\_\_\_\_\_\_\_\_\_\_

%signal reÃ§u(complet)(permet de voir graphiquement le fonctionnement de la BLU)

figure(1);

subplot(211);

plot(t,Signal);

legend('Signal reçu en temporel');

subplot(212);

[spectre4,f] = TransFourier(Signal,t);

plot(f,spectre4);

legend('Signal reçu en fréquentiel');

%\_\_\_\_\_\_\_\_\_\_

%Premier signal

figure(2);

[spectre1,f] = TransFourier(signalDemodule2,t);

subplot(211);

plot(f,abs(spectre1));

legend('Signal 1 démodule et filtre');

%\_\_\_\_\_\_\_\_\_\_

%Deuxième signal

figure(3);

[spectre2,f]=TransFourier(signalSomme, t);

subplot(211);

plot(f,spectre2);

legend('Signal 2 demodule et filtre');