Communications numériques

Travaux Pratiques (15 h)

Modélisation et étude d'une chaîne de communication numérique

Université Paris 13, Institut Galilée, Ecole d'ingénieurs Sup Galilée
Parcours Informatique et Réseaux Apprentissage - 2ème année

2018-2019

Consignes

- Récupérer le fichier `TP.zip` sur le site

  `http://www.laurentoudre.fr/comnum.html`

- Ouvrir MATLAB et créer un répertoire de travail. Dézipper le fichier `TP.zip` dans ce répertoire.
- A la fin de la séance, récupérer les scripts que vous avez écrits et vous les envoyer par e-mail afin de les conserver pour la prochaine séance.

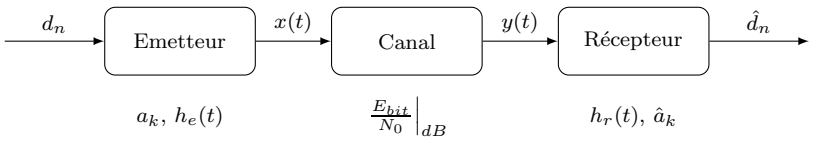
Plan de l'étude

1 Transmission en bande de base **2**
1.1 Emetteur . . . 2
1.1.1 Conversion bits/symboles . . . 2
1.1.2 Génération du peigne de Dirac . . . 3
1.1.3 Filtre de mise en forme . . . 4
1.1.4 Energie moyenne par bit . . . 5
1.1.5 Finalisation de l'émetteur . . . 5
1.1.6 Propriétés des signaux en bande de base . . . 5
1.2 Canal . . . 6
1.3 Récepteur . . . 6
1.3.1 Filtre de réception . . . 6
1.3.2 Echantillonnage . . . 7
1.3.3 Décision . . . 7
1.3.4 Décodage . . . 7
1.3.5 Finalisation du récepteur . . . 7
1.3.6 Performances de la chaine de communication . . . 8

2 Transmission en bande modulée **8**
2.1 Emetteur . . . 9
2.2 Canal . . . 9
2.3 Récepteur . . . 10

1 Transmission en bande de base

Le but de cette partie est d'étudier une chaîne de transmission en bande de base en présence de bruit blanc gaussien et illustrée ci-dessous.



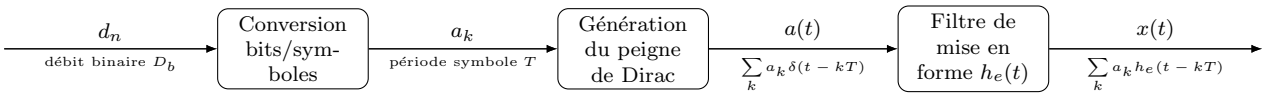
- A l'émetteur, on part d'une série de bits d_n, que l'on convertit en symboles a_k selon un dictionnaire choisi. On crée ensuite à partir de ces symboles un signal physique $x(t)$ grâce à un filtre de mise en forme $h_e(t)$ correctement choisi
- Le canal est modélisé uniquement par un bruit additif gaussien $b(t)$ de variance $\sigma^2 = \dfrac{N_0}{2}$, créant un signal bruité $y(t) = x(t) + b(t)$.
- Au niveau du récepteur, on aura un filtre de réception $h_r(t)$ (choisi pour que le récepteur soit optimal), une étape d'échantillonnage puis de prise de décision pour retrouver les symboles $\hat{a}_k$. Enfin, on décodera les symboles pour tenter de retrouver le message binaire original $\hat{d}_n$.

Deux points fondamentaux seront étudiés dans cette partie :

- **L'étude des propriétés du signal en bande de base.** On étudiera en particulier l'influence des dictionnaires et des filtres de mise en forme sur les propriétés énergétiques et spectrales du signal ainsi construit (bande passante, atténuations, annulations....).
- **L'étude détaillée des performances de la chaîne de communication** en terme notamment de taux d'erreur binaire. On étudiera l'influence des filtres d'émission et de réception, le choix du dictionnaire, ainsi que le rapport signal sur bruit $\frac{E_{bit}}{N_0}\Big|_{dB}$ sur les performances de la chaîne de traitement.

1.1 Emetteur

Le plan de l'émetteur est détaillé sur le schéma bloc ci-dessous :



Les différentes étapes à réaliser, illustrées ci-dessus, sont les suivantes :

1. Conversion grâce à un dictionnaire du message binaire d_n (émis avec un débit binaire D_b) en une série de symboles a_k (période symbole T)
2. Construction du signal physique $a(t) = \sum_k a_k \delta(t - kT)$ (un symbole émis toutes les T secondes)
3. Convolution du signal $a(t)$ par la réponse impulsionnelle $h_e(t)$ du filtre d'émission, pour former $x(t)$

1.1.1 Conversion bits/symboles

Le but de l'étape de conversion bits/symboles est d'associer à chaque groupement de m bits un symbole pris dans un dictionnaire à $M = 2^m$ éléments. On s'intéressera dans ce TP uniquement aux dictionnaires M-aire antipolaires avec $M = 2^m$ élements :

$$a_k \in \{-(M-1), \cdots, -3, -1, 1, 3, \cdots, M-1\} \text{ (uniquement les éléments impairs)}$$

Q1 - Créer une fonction MATLAB

```
[ak,T,K] = conversion_bitssymboles(dn,Db,M,method_mod)
```

prenant en entrée une série de bits `dn` émise avec un débit binaire `Db` et formant une série de `K` symboles `ak` associés à une période symbole `T`. Les variables `M` et `method_mod` permettent de sélectionner le dictionnaire de symboles à utiliser.

Indications

- Il faudra d'abord vérifier que `M` est bien une puissance de 2, calculer $m = \log_2(M)$, puis la période symbole `T` en fonction du débit binaire `Db` et de la valence `M`.
- On utilisera la fonction `[dico_sym,dico_dec]=dictionnaire(M,method_mod)` fournie, prenant en entrée la valence `M` et le type de dictionnaire `method_mod`, et retournant la liste des symboles du dictionnaire `dico_sym` et la représentation décimale des messages binaires associés à chaque symbole `dico_dec`. Exemple : si le symbole -7 est associé au message 0 1 1, alors on stockera -7 dans le vecteur `dico_sym` et la représentation décimale 3 dans le vecteur `dico_dec`.
- On traitera un par un les groupes de $m = \log_2(M)$ bits : on commencera par les représenter en base 10 (grâce à la fonction `bits2dec.m` fournie), puis on cherchera dans le dictionnaire le symbole associé à ce nombre décimal

Tests à effectuer

Pour tester la fonction, on pourra générer un signal binaire aléatoire de longueur `N` = 10 grâce à la ligne de code suivante :

```
d = round (rand(N,1));
```

Dans cette première partie du TP on travaille sur des signaux en bande de base, il faut donc choisir `method_mod='BdB'`. On utilisera un débit binaire `Db` = 2 bits/sec. On testera trois valeurs pour `M` : 2, 4 et 8 et on comparera, pour les 3 configurations, les résultats obtenus (symboles + période symbole) avec les résultats théoriques trouvés à la main.

Q2 - Créer une fonction MATLAB

```
[dn] = conversion_symbolesbits(ak,M,method_mod)
```

prenant en entrée une série de symboles `ak` et formant la série de bits `d` initiale. Les variables `M` et `method_mod` permettent de sélectionner le dictionnaire de symboles à utiliser.

Indications

On commencera par chercher la représentation en base 10 du symbole, que l'on convertira ensuite en message binaire grâce à la fonction `dec2bits.m` fournie.

Tests à effectuer

Si tout fonctionne, lorsqu'on convertit le message binaire en symboles puis les symboles en message binaire, on doit retrouver le message original. Tester que ceci est vrai pour les 3 configurations avec des messages binaires aléatoires de taille `N` = 10 et un débit binaire `Db` = 2 bits/sec.

1.1.2 Génération du peigne de Dirac

Q3 - Créer une fonction

```
[a,t_a]=genere_dirac(ak,K,T,Fs)
```

prenant en entrée une série de `K` symboles `ak` avec une période symbole `T`, et générant le signal physique `a` associé à un vecteur temps `t_a` et échantillonné à la fréquence d'échantillonnage `Fs`.

Indications

- Commencer par déterminer la durée `Nsym` (en échantillons) d'un symbole à partir de `T` et `Fs`
- Calculer le vecteur temps `t_a` à utiliser. Attention, ce vecteur doit contenir $K \times N_{sym}$ échantillons.
- Déterminer les indices du vecteur `t_a` correspondant aux temps multiples de `T`
- Définir ensuite le signal `a`, de même taille que le vecteur temps `t_a`, et former le signal

$$a(t) = \sum_{k=0}^{K-1} a_k \delta(t - kT)$$

Tests à effectuer

Générer un signal binaire aléatoire de longueur `N` = 10 émis avec un débit binaire `Db` = 2 bits/sec, et le convertir en symboles grâce au dictionnaire de votre choix. Générer et afficher ensuite en fonction du temps le signal $a(t)$ obtenu. On prendra `Fs` = 100 Hz.

1.1.3 Filtre de mise en forme

Q4 - Créer une fonction MATLAB

```
[he,E_he]=create_filter(t_he,T,method_fil)
```

prenant en entrée un vecteur temps `t_he` et la période symbole `T` et formant la réponse impulsionnelle `he` d'un filtre de mise en forme. La variable `method_fil` permet de sélectionner le filtre que l'on veut générer et `E_he` représente l'énergie totale du filtre.

- `method_fil='NRZ'` : filtre NRZ

$$h_e(t) = \begin{cases} 1 & \text{si } 0 \leq t < T \\ 0 & \text{sinon} \end{cases}$$

- `method_fil='RZ'` : filtre RZ

$$h_e(t) = \begin{cases} 1 & \text{si } 0 \leq t < \frac{T}{2} \\ 0 & \text{sinon} \end{cases}$$

- `method_fil='biphase'` : filtre biphase Manchester

$$h_e(t) = \begin{cases} 1 & \text{si } 0 \leq t < \frac{T}{2} \\ -1 & \text{si } \frac{T}{2} \leq t < T \\ 0 & \text{sinon} \end{cases}$$

- `method_fil='RCS'` : filtre en racine de cosinus surélevé

Indications

- Pour des raisons de précision numérique, il se peut que les inégalités strictes soient mal comprises par MATLAB. Il est donc préférable d'écrire $a < b$ comme $a < b - \epsilon$ avec par exemple $\epsilon = 10^{-10}$.
- L'énergie totale du filtre sera estimée comme la somme des éléments de la réponse impulsionnelle au carré.

$$E_{h_e} = \sum_n |h_e[n]|^2$$

- Pour réaliser le filtre en racine de cosinus surélevé, on utilisera la fonction `[he]=rootraisedcosine(t,T,beta)` fournie. Dans la suite on supposera que le paramètre du filtre est $\beta = 0.25$ (on pourra le changer par la suite si on le désire).

Tests à effectuer

Pour les tests, on pourra utiliser `T` = 0.5 secondes. Définir un vecteur temps `t_he` compris entre $-5T$ et $5T$ et échantillonné à `Fs` = 100 Hz. Tracer la réponse impulsionnelle des 4 filtres définis ci dessus en fonction du temps.

Q5 - Créer une fonction MATLAB

```
[x,t_x,E_he]=filtre_emission(a,t_a,T,Fs,method_fil)
```

qui prend en entrée un signal physique `a` (somme de dirac), le convolue avec un filtre de mise en forme spécifié par la variable `method_fil` et sort un signal `x`. `t_a` et `t_x` sont les vecteurs temps respectivement associés à `a` et `x` et `Fs` est la fréquence d'échantillonnage de ces deux signaux physiques. `T` est la période symbole. `E_he` est ici l'énergie du filtre d'émission utilisé.

Indications

- Pour réaliser la convolution entre le signal $a(t)$ et la réponse impulsionnelle du filtre de mise en forme $h_e(t)$ afin de former $x(t)$, on utilisera la fonction `[y,ty]=prodconv(x1,t1,x2,t2,Fs)` fournie.
- Pour cela, on considèrera des réponses impulsionnelles $h_e(t)$ du filtre pour des temps compris entre $-5T$ et $5T$.
- Attention, le signal obtenu en sortie n'a plus le même support temporel, il faut donc travailler ensuite avec le nouveau vecteur temps fourni en sortie de la fonction `[y,ty]=prodconv(x1,t1,x2,t2,Fs)`.

Tests à effectuer

Voir question 7.

1.1.4 Energie moyenne par bit

Q6 - Créer une fonction MATLAB

```
[E_bit]=energie_moyenne_bit(M,method_mod,E_he)
```

calculant l'énergie moyenne par bit `E_bit` pour un dictionnaire spécifié par `M` et `method_mod`, et un filtre de mise en forme d'énergie totale `E_he`.

Indications

On utilisera pour cela la formule théorique du cours

$$E_{bit} = \frac{1}{M \log_2 M} \sum_{i=1}^{M} |a_i|^2 E_{h_e}$$

et la fonction `[dico_sym,dico_dec]=dictionnaire(M,method_mod)` fournie.

Tests à effectuer

On vérifiera en prenant `E_he` = 1 que l'on retrouve bien les résultats théoriques pour les 3 dictionnaires considérés.

1.1.5 Finalisation de l'émetteur

Q7 - En utilisant les fonctions MATLAB précédemment codées, créer une fonction

```
[x,t_x,ak,K,T,E_bit]=emetteur(dn,Db,Fs,M,method_fil)
```

générant un signal `x` mis en forme à partir d'une série de bits `dn`. `t_x` représente le vecteur temps associé au signal `x`, `T` la période symbole, `ak` la série de `K` symboles et `E_bit` l'énergie moyenne par bit. `Db` représente ici le débit binaire, `Fs` la fréquence d'échantillonnage. `M` spécifie le dictionnaire à utiliser et `method_fil` le filtre de mise en forme à utiliser.

Indications

L'émetteur réalisera successivement la conversion bits/symboles, la génération du peigne de Dirac, le calcul de la réponse impulsionnelle du filtre, la convolution entre le signal $a(t)$ et la réponse impulsionnelle du filtre $h_e(t)$, et le calcul de l'énergie moyenne par bit.

Tests à effectuer

Générer aléatoirement un signal binaire de taille `N` = 10, émis à un débit binaire `Db` = 2 bits/seconde. Tracer en fonction du temps le signal en bande de base `x` obtenu pour tous les choix de dictionnaire et de filtres d'émission. Les signaux physiques seront échantillonnés à `Fs` = 100 Hz. Attention, à cause du produit de convolution, le vecteur temps `t_x` est défini pour des temps inférieurs à 0 et supérieurs à KT. On utilisera donc la commande `xlim([0 K*T])` de MATLAB pour le tracer uniquement sur la période souhaitée.

1.1.6 Propriétés des signaux en bande de base

Q8 - Tracer la densité spectrale de puissance (en échelle linéaire et logarithmique) obtenue avec un dictionnaire binaire antipolaire et un filtre NRZ. Le signal est-il en bande de base ? Estimer visuellement sa largeur de bande.

Indications

Générer aléatoirement un signal binaire de taille `N` = 24000, émis à un débit binaire `Db` = 2 bits/seconde, et avec `Fs` = 100 Hz. On utilisera la fonction `compute_dsp(x,Fs,method_dsp)` fournie pour estimer la DSP, en prenant garde d'estimer cette DSP uniquement sur les temps pertinents.

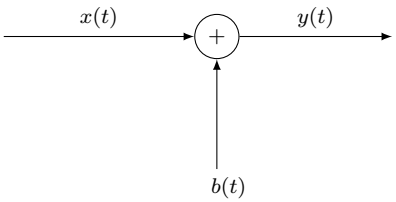
```
compute_dsp(x(t_x>=0 & t_x<K*T),Fs,'log');

compute_dsp(x(t_x>=0 & t_x<K*T),Fs,'lin');
```

Q9 - Reproduire la même expérience toujours avec un dictionnaire binaire antipolaire mais en faisant varier le filtre de mise en forme. Estimer visuellement la largeur de bande pour chacune des configurations. Le choix du filtre de mise en forme influence-t-il la largeur de bande ?

Q10 - Reproduire la même expérience avec un filtre NRZ mais en faisant varier cette fois le dictionnaire utilisé (M = 2, 4 et 8). Estimer visuellement la largeur de bande pour chacune des configurations. La valence du dictionnaire influence-t-elle la largeur de bande ?

1.2 Canal

Le plan du canal de transmission est détaillé sur le schéma bloc ci-dessous :



Le canal est modélisé uniquement par un bruit additif gaussien $b(t)$ créant un signal bruité $y(t) = x(t) + b(t)$.

Q11 - La fonction modélisant le canal est fournie. Cette fonction

```
[y,t_y]=canal(x,t_x,E_bit,EbitN0_dB);
```

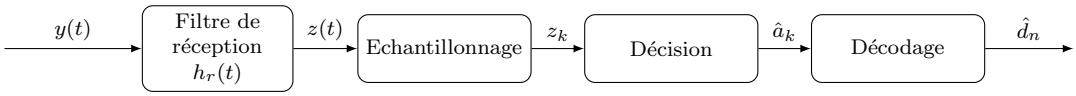
prend en entrée le signal `x` et lui rajoute un bruit blanc additif gaussien dont la variance est calculée à partir du rapport signal sur bruit `EbitN0_dB` en décibels et de l'énergie moyenne par bit `E_bit`

Tests à effectuer

On pourra visualiser les signaux bruités obtenus avec des valeurs de `EbitN0_dB` égale à 1000 (pas de bruit) et 0 (niveau de bruit élevé). Pour les simulations, on prendra `N` =10, `Db` = 2 bits/seconde, et `Fs` = 100 Hz, et on génèrera un signal $x(t)$ avec un dictionnaire et un filtre de mise en forme au choix. On utilisera la commande `xlim([0 K*T])` de MATLAB pour le tracer uniquement sur la période souhaitée.

1.3 Récepteur

Le plan du récepteur est détaillé sur le schéma bloc ci-dessous :



Les différentes étapes à réaliser, illustrées ci-dessus, sont les suivantes :

1. Convolution du signal $y(t)$ par la réponse impulsionnelle $h_t(t)$ du filtre de réception, pour former $z(t)$
2. Echantillonnage du signal $z(t)$ pour tous les temps $t = kT$ avec $k \in [\![0, K-1]\!]$, afin de récupérer z_k
3. Décision pour estimer les symboles $\hat{a}_k$ à partir des z_k et de E_{he}
4. Décodage pour retrouver le message binaire $\hat{d}_n$ à partir des symboles estimés $\hat{a}_k$

On supposera ici que le récepteur est optimal (rappel : on a donc $h_r(t) = h_e(-t)$ et $h = h_e * h_r$ qui est un filtre de Nyquist).

1.3.1 Filtre de réception

Q12 - Créer une fonction MATLAB

```
[z,t_z,E_hr]=filtre_reception(y,t_y,T,Fs,method_fil)
```

qui prend en entrée un signal bruité `y`, le convolue avec un filtre de réception spécifié par la variable `method_fil` et sort un signal `z`. `t_y` et `t_z` sont les vecteurs temps respectivement associés à `y` et `z` et `Fs` la fréquence d'échantillonnage. `T` est la période symbole et `E_hr` est ici l'énergie du filtre d'émission h_r utilisé (qui est aussi égale à celle du filtre d'émission).

Indications

- Pour créer la réponse impulsionnelle du filtre de réception, il suffit de créer celle du filtre d'émission mais avec $t = -t$
- On prendra toujours la réponse impulsionnelle du filtre entre $-5T$ et $5T$
- Attention, la convolution change le vecteur de temps donc `t_z` n'est pas le même que `t_y`.

Tests à effectuer

On pourra visualiser le signal $z(t)$ obtenu avec différentes valeurs de `EbitN0_dB`. Pour les simulations, on prendra `N` =10, `Db` = 2 bits/seconde, et `Fs` = 100 Hz, et on génèrera un signal $x(t)$ avec un dictionnaire et un filtre de mise en forme au choix. Faire varier le filtre de mise en forme et observer les conséquences sur le signal $z(t)$.

1.3.2 Echantillonnage

Q13 - Créer une fonction MATLAB

```
[zk]=echantillonnage(z,t_z,T,K,Fs)
```

qui prend un signal `z` échantilonné à `Fs` et associé à un vecteur temps `t_z`, et renvoie un vecteur `zk` de longueur `K`, stockant les valeurs de `z` correspondant aux temps $t = kT$ avec $k \in [\![0, K-1]\!]$.

Indications

- Dans un premier temps, il s'agit de calculer la durée `Nsym` (en échantillons) d'un symbole
- On cherchera ensuite l'indice du vecteur `t_z` correspondant au temps $t = 0$ grâce à la commande `find` de MATLAB.
- A partir de là, et en s'inspirant de ce qui a été fait pour la fonction `genere_dirac`, il s'agit de trouver les indices du vecteur `t_z` correspondant aux temps multiples de `T`. Attention, il faut s'arrêter à $t = (K-1)T$!)

Tests à effectuer

Tester la fonction sur un des signaux $z(t)$ généré à la question précédente et vérifier en comparant à la figure que l'on a bien échantillonné aux bons instants.

1.3.3 Décision

Q14 - Créer une fonction

```
[ak_hat]=decision(zk,E_hr,M,method_mod)
```

qui prend une série de valeurs `zk` et associe à chacune un symbole `ak_hat` du dictionnaire spécifié par `M` et `method_mod`. `E_hr` est ici l'énergie du filtre de réception h_r utilisé.

Indications

Comme nous l'avons vu dans le cours, étant donnée une valeur de z_k, le symbole détecté est le symbole du dictionnaire le plus proche de la valeur $\frac{z_k}{E_{he}}$ (au sens de la distance euclidienne).

Tests à effectuer

On pourra vérifier qu'en absence de bruit, on retrouve exactement les symboles a_k.

1.3.4 Décodage

Cette partie a déjà été réalisée, il s'agit de la fonction `conversion_symbolesbits` écrite dans la question Q2.

1.3.5 Finalisation du récepteur

Q15 - En utilisant les fonctions MATLAB précédemment codées, créer une fonction

```
[dn_hat,ak_hat]=recepteur(y,t_y,T,K,Fs,M,method_fil)
```

qui prend en entrée un signal bruité `y` (échantillonné à `Fs` et associé à un vecteur temps `t_y`) et renvoyant une estimation du message binaire originellement envoyé `dn_hat`. `ak_hat` représente l'estimation des symboles envoyés, `T` est la période symbole, `K` le nombre de symboles à retrouver, `M` et `method_fil` précisent respectivement le dictionnaire et les filtres à utiliser.

Indications

Le récepteur réalisera successivement la convolution par le filtre de réception, l'échantillonnage, la décision et le décodage.

Tests à effectuer

Pour les simulations, on prendra comme toujours `N` =10, `Db` = 2 bits/seconde, et `Fs` = 100 Hz. Si tout fonctionne bien, en prenant $y = x$ (absence de bruit), on doit retrouver exactement les bits envoyées.

1.3.6 Performances de la chaine de communication

Q16 - Estimer le taux d'erreur binaire

$$TEB = \frac{\text{nombre de bits mal transmis}}{\text{nombre total de bits emis}}$$

avec un filtre NRZ et un dictionnaire binaire antipolaire pour des valeurs de rapport signal sur bruit $\left.\frac{E_{bit}}{N_0}\right|_{dB}$ comprises entre 0 et 5 dB. Tracer la courbe du taux d'erreur binaire en fonction du rapport $\left.\frac{E_{bit}}{N_0}\right|_{dB}$ et comparer aux résultats théoriques.

Indications

- Générer des signaux aléatoires avec `N` = 24000, `Db` = 2 bits/seconde et `Fs` = 100 Hz
- Evaluer le taux d'erreur binaire grâce à la commande

```
TEB = sum(d~=d_hat)/length(d);
```

- Pour tracer la courbe d'erreur avec une échelle logarithmique on peut utiliser la commande

```
semilogy(EbitN0_dB,TEB);
```

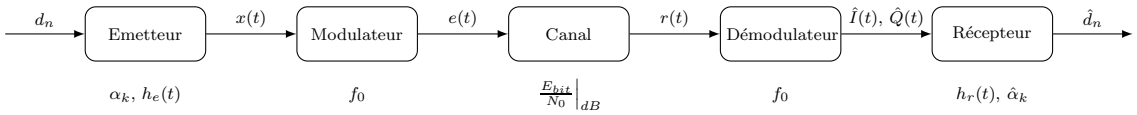
- On utilisera la fonction `TEB_theo=TEB_theorique(EbitN0_dB,M,method_mod)` fournie pour connaître les taux d'erreur binaire théoriques.

Q17 - Reproduire la même expérience toujours avec un dictionnaire binaire antipolaire mais en faisant varier le filtre de mise en forme. Comparer les courbes expérimentales et théoriques. Le taux d'erreur binaire dépend-il du filtre ?

Q18 - Reproduire la même expérience avec un filtre NRZ mais en faisant varier cette fois le dictionnaire utilisé ($M = 2$, 4 et 8). Comparer les courbes expérimentales et théoriques. Le taux d'erreur binaire dépend-il de la valence M ?

2 Transmission en bande modulée

Le but de cette partie est d'étudier une chaîne de transmission en bande modulée en présence de bruit blanc gaussien et illustrée ci-dessous.



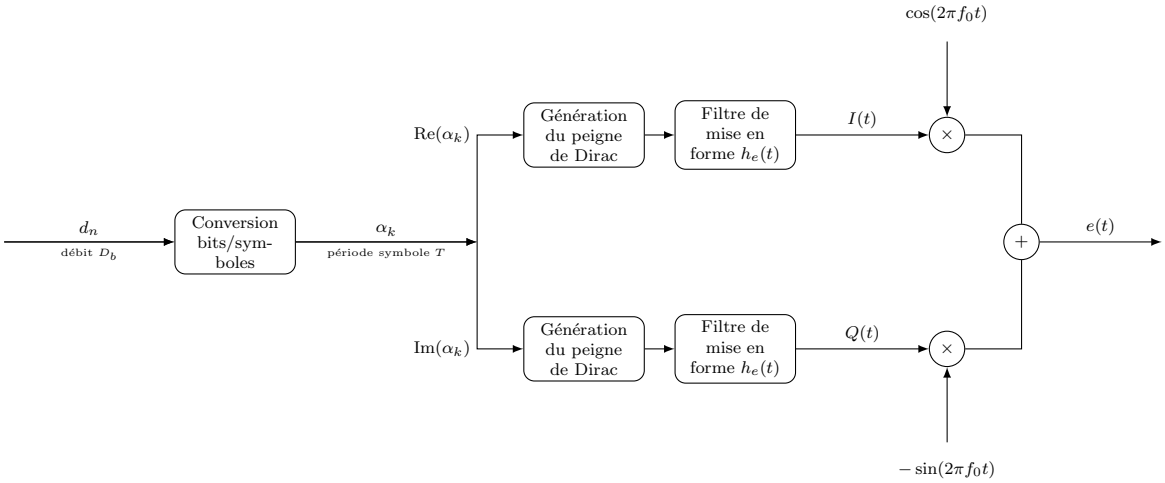
- A l'émetteur, on part d'une série de bits d_n, que l'on convertit en symboles complexes α_k selon le type de modulation choisi. On crée ensuite à partir de ces symboles un signal en bande de base $x(t)$ grâce à un filtre de mise en forme h_e correctement choisi. La partie réelle de ce message $I(t)$ est envoyée sur une porteuse en phase $\cos(2\pi f_0 t)$, et la partie imaginaire $Q(t)$ sur une poteuse en quadrature de phase $-\sin(2\pi f_0 t)$ pour former un signal $e(t)$.
- Le canal est modélisé uniquement par un bruit additif gaussien $b(t)$ de variance σ^2, créant un signal bruité $r(t) = e(t) + b(t)$.
- Au niveau du récepteur, on aura tout d'abord une étape de démodulation et de filtrage pour récupérer les composantes en phase et en quadrature de phase $\hat{I}(t)$ et $\hat{Q}(t)$. Ces composantes passeront par un filtre de réception $h_r(t)$ (adapté à $h_e(t)$), par une étape d'échantillonnage puis de prise de décision pour retrouver les symboles complexes $\hat{\alpha}_k$. Enfin, on décodera les symboles $\hat{\alpha}_k$ pour tenter de retrouver le message binaire original $\hat{d}_n$.

On supposera ici que le récepteur est optimal, ce qui guidera le choix des filtres d'émission et de réception (rappel : on doit avoir $h_r(t) = h_e(-t)$ et $h = h_e * h_r$ qui doit être un filtre de Nyquist).

On étudiera **uniquement des filtres d'émission NRZ** dans cette partie, et on étudiera l'influence du type de modulation sur les performances de la chaîne de transmission.

2.1 Emetteur

Le plan de l'émetteur est détaillé sur le schéma bloc ci-dessous :



Les différentes étapes à réaliser, illustrées ci-dessus, sont les suivantes :

1. Conversion grâce à un dictionnaire du message binaire d_n (émis avec un débit binaire D_b) en une série de symboles complexes α_k (période symbole T)
2. Création de la composante en phase $I(t)$ transmettant la partie réelle des symboles α_k. Création de la composante en quadrature de phase $Q(t)$ transmettant la partie imaginaire des symboles α_k.
3. Génération du signal modulée $e(t)$ grâce à des porteuses de fréquence f_0.

Q19 - En utilisant les fonctions MATLAB précédemment codées, créer une fonction

```
[e,t_e,alphak,K,T,E_bit,I,Q]=emetteur_mod(dn,Db,Fs,M,method_mod,method_fil,f0)
```

générant un signal `e` modulé avec une fréquence fondamentale `f0` à partir d'une série de bits `dn`. `t_e` représente le vecteur temps associé au signal `e`, `alphak` la liste des `K` symboles complexes qui ont été transmis, `T` la période symbole, `E_bit` l'énergie moyenne par bit. `Db` représente ici le débit binaire, `Fs` la fréquence d'échantillonnage, `M` la taille du dictionnaire, `method_mod` le type de modulation à utiliser et `method_fil` le filtre de mise en forme à utiliser. `I` et `Q` sont respectivement les composantes en phase et en quadrature de phase

Indications

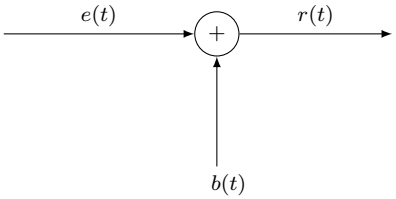
L'émetteur réalisera successivement toutes les étapes décrites dans le schéma ci-dessus.

Tests à effectuer

On pourra tester la fonction avec un message binaire de taille `N` = 12, un débit binaire `Db` = 2 bits/seconde, une fréquence d'échantillonnage `Fs` = 100 Hz, et une fréquence fondamentale `f0` = 13 Hz. On observera les signaux $e(t)$, $I(t)$ et $Q(t)$ obtenus pour différentes modulations.

2.2 Canal

Le plan du canal de transmission est détaillé sur le schéma bloc ci-dessous :



Q20 - La fonction modélisant le canal est fournie. Cette fonction

```
[r,t_r]=canal_mod(e,t_e,E_bit,EbitN0_dB);
```

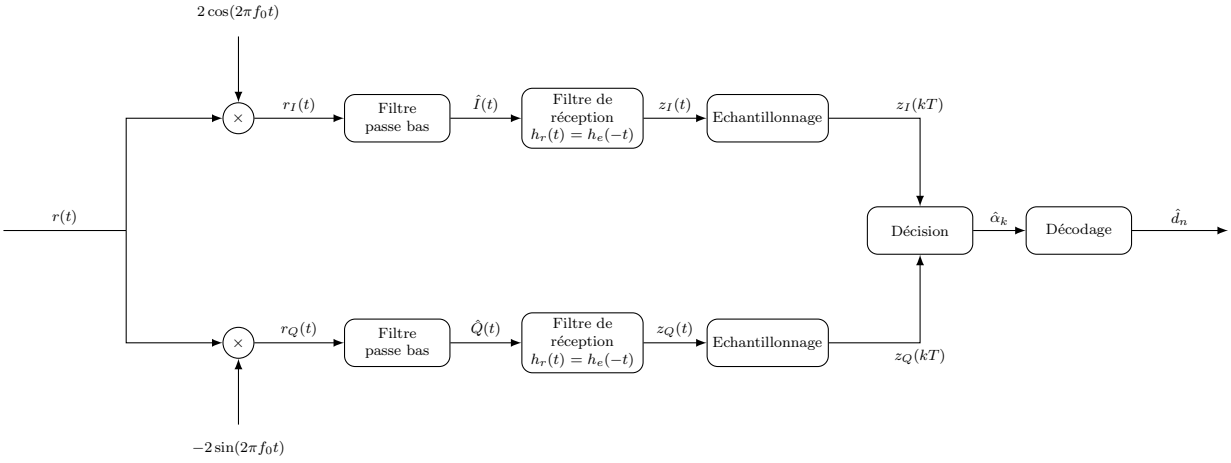
prend en entrée le signal `e` et lui rajoute un bruit blanc additif gaussien dont la variance est calculée à partir du rapport signal sur bruit `EbitN0_dB` en décibels et de l'énergie moyenne par bit `E_bit`

Tests à effectuer

On pourra visualiser les signaux bruités obtenus avec des valeurs de `EbitN0_dB` égale à 1000 (pas de bruit) et 0 (niveau de bruit élevé). Pour les simulations, on prendra `N` =12, `Db` = 2 bits/seconde, et `Fs` = 100 Hz, et on génèrera un signal $x(t)$ avec une modulation et un filtre de mise en forme au choix. On utilisera la commande `xlim([0 K*T])` de MATLAB pour le tracer uniquement sur la période souhaitée.

2.3 Récepteur

Le plan du récepteur est détaillé sur le schéma bloc ci-dessous :



Les différentes étapes à réaliser, illustrées ci-dessus, sont les suivantes :

1. Multiplier le signal $r(t)$ respectivement par $2\cos(2\pi f_0 t)$ et $-2\sin(2\pi f_0 t)$ pour obtenir les signaux $r_I(t)$ et $r_Q(t)$
2. Filtrer ces signaux grâce à un filtre passe-bas pour obtenir des estimées $\hat{I}(t)$ et $\hat{Q}(t)$
3. Convolution des signaux $\hat{I}(t)$ et $\hat{Q}(t)$ par la réponse impulsionnelle $h_t(t)$ du filtre de réception, pour former $z_I(t)$ et $z_Q(t)$
4. Echantillonnage des signaux $z_I(t)$ et $z_Q(t)$ pour tous les temps $t = kT$ avec $k \in [\![0, K-1]\!]$, afin de récupérer $z_I(kT)$ et $z_Q(kT)$
5. Décision pour estimer les symboles $\hat{\alpha}_k$ à partir des $z_I(kT)$, $z_Q(kT)$ et de E_{h_e}
6. Décodage pour retrouver le message binaire $\hat{d}_n$ à partir des symboles estimés $\hat{\alpha}_k$

On supposera ici que le récepteur est optimal (rappel : on a donc $h_r(t) = h_e(-t)$ et $h = h_e * h_r$ qui est un filtre de Nyquist).

Q21 - En utilisant les fonctions MATLAB précédemment codées, créer une fonction

```
[dn_hat,alphak_hat,zk]=recepteur_mod(r,t_r,T,K,Fs,M,method_fil,method_mod,f0)
```

qui prend en entrée un signal `r` échantillonné à `Fs`, modulé par une porteuse de fréquence `f0` et bruité (auquel correspond un vecteur temps `t_r`) et renvoyant une estimation du message binaire originellement envoyé `dn_hat`. `alphak_hat` représente l'estimation des symboles complexes envoyés et `z_k` les symboles complexes obtenus à la sortie du récepteur avant décision. `T` est la période symbole, `K` le nombre de symboles à retrouver, `M` et `method_mod` les paramètres de la modulation et `method_fil` le type de filtre à utiliser.

Indications

- Pour la prise de décision, il faudra, selon le type de modulation, soit utiliser des symboles réels $z_k = z_I(kT)$ (modulations ASK et BPSK), soit des symboles complexes $z_k = z_I(kT) + jz_Q(kT)$ (toutes les autres modulations)
- Le filtrage passe-bas pourra être fait grâce à la commande suivante :

```
[b,a] = butter(5,f0*2/Fs);

y = filtfilt(b,a,x);
```

Tests à effectuer

On pourra tester la fonction avec un message binaire à `N` = 12 bits, un débit binaire `Db` = 2 bits/seconde, une fréquence d'échantillonnage `Fs` = 100 Hz, et une fréquence fondamentale `f0` = 13 Hz. Si tout fonctionne bien, en prenant $r = e$, on doit retrouver exactement les bits envoyées.

Q22 - Choisir 3 modulations parmi les 8 proposées et regarder dans le plan complexe les symboles z_k obtenus pour un rapport signal sur bruit $\left.\frac{E_{bit}}{N_0}\right|_{dB}$ de 0 et 10 dB et commenter.

Indications

- Pour cela, on génèrera des signaux aléatoires avec `N` = 24000, `Db` = 2 bits/seconde, `Fs` = 100 Hz et `f0` = 13 Hz.
- On pourra utiliser la commande MATLAB

```
plot(real(z_k),imag(z_k),'.')
```

Q23 - Choisir 3 modulations parmi les 8 proposées et étudier les densités spectrales de puissance des signaux modulés. Comparer à ce qui se passe en bande de base.

Indications

- Pour cela, on génèrera des signaux aléatoires avec `N` = 24000, `Db` = 2 bits/seconde, `Fs` = 100 Hz et `f0` = 13 Hz.
- On utilisera la fonction `compute_dsp(x,Fs,method_dsp)` fournie pour estimer la DSP, en prenant garde d'estimer cette DSP uniquement sur les temps pertinents.

Q24 - Choisir 3 modulations parmi les 8 proposées et tracer la courbe du taux d'erreur binaire en fonction du rapport $\left.\frac{E_{bit}}{N_0}\right|_{dB}$ pour des valeurs de rapport signal sur bruit $\left.\frac{E_{bit}}{N_0}\right|_{dB}$ comprises entre 0 et 5 dB. Comparer aux courbes théoriques.

Indications

- Pour cela, on génèrera des signaux aléatoires avec `N` = 24000, `Db` = 2 bits/seconde, `Fs` = 100 Hz et `f0` = 13 Hz.
- On utilisera la fonction `TEB_theo=TEB_theorique(EbitN0_dB,M,method_mod)` fournie pour connaître les taux d'erreur binaire théoriques.