TS111

Simulation d'un émetteur / récepteur ADS-B

Table des matières

I- Introduction au système de diffusion ADS-B

Afin de surveiller l'état du réseau aérien, un système de diffusion appelé Automatic Dependant Surveillance - Broadcast (ADS-B) a été proposé en complément des radars classiques. Dans ce système, les appareils estiment leurs positions (longitude, latitude, altitude) grâce aux techniques de positionnement par satellite (GPS - Global Positionning System, GLONASS - Global Navi- gation Satellite System ou encore Galiléo) et diffusent ces informations régulièrement (toutes les secondes environ). Ces informations sont ensuite récupérées :

- au sol : par des stations intermédiaires ou des tours de contrôle,
- dans les autres appareils : qui peuvent utiliser ces signaux pour leurs systèmes anticollision.

Le principal avantage du système ADS-B par rapport au radar classique est son faible coût d'infrastructure. En effet, la station réceptrice possède seulement une antenne de réception afin de recevoir les signaux ADS-B, le reste des traitements étant faits à bord des appareils. Le récepteur est donc entièrement passif et n'a pas besoin d'interroger l'appareil pour qu'il émette sa position.
Il existe plusieurs liaisons pour la transmission des signaux ADS-B. Les deux principales sont :

- le 1090 Extended Squitter (1090 ES),
- l'UAT (Universal Access Transponder).

Dans ce projet, nous allons nous concentrer sur le 1090 ES car c'est le mode privilégié en Europe. Dans l'appellation 1090 ES, 1090 signifie que la fréquence porteuse des signaux ADS-B (pour ce mode de transmission) est 1090 MHz. ES signifie Extended Squitter, ce qui pourrait être traduit par message étendu. En effet, les messages transmis en utilisant ce mode avaient une durée de 56 bits (hors préambule de synchronisation). Dans le nouveau standard, les messages peuvent contenir 112 bits, d'où la notion de "message étendu".

II- Étude préliminaire

a. La chaîne de communication

Le but de ce projet va être d'implémenter la chaîne de communication du système de diffusion ADS-B afin d'être en mesure de traiter et décoder des signaux réels provenant des avions par la suite.

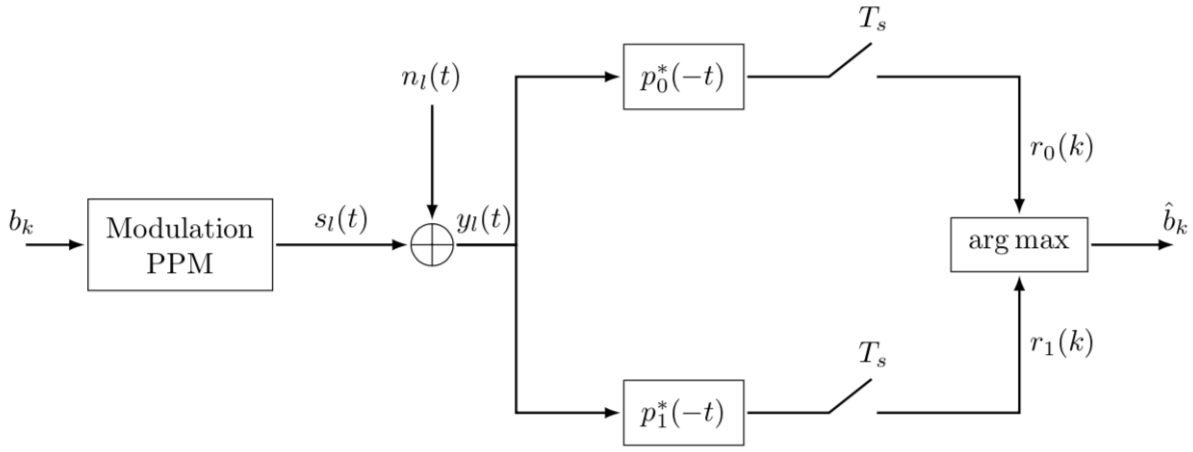


Figure 1. Chaîne de communication complète du système ADS-B

La chaîne de communication à implémenter se trouve en figure 1. Chacun des blocs va être décrit et implémenté sous Matlab. Il faudra également prévoir un bloc de synchronisation en temps et en fréquence afin de détecter le début de la séquence à traiter.

Question 1 : En partant de la suite binaire « 011010 », et en considérant que le bruit $n_{l(t)} = 0$, les signaux $s_l(t)$, $p_0^*(-t)$ et $p_1^*(-t)$ sont respectivement représentés en figure 2 et 3.

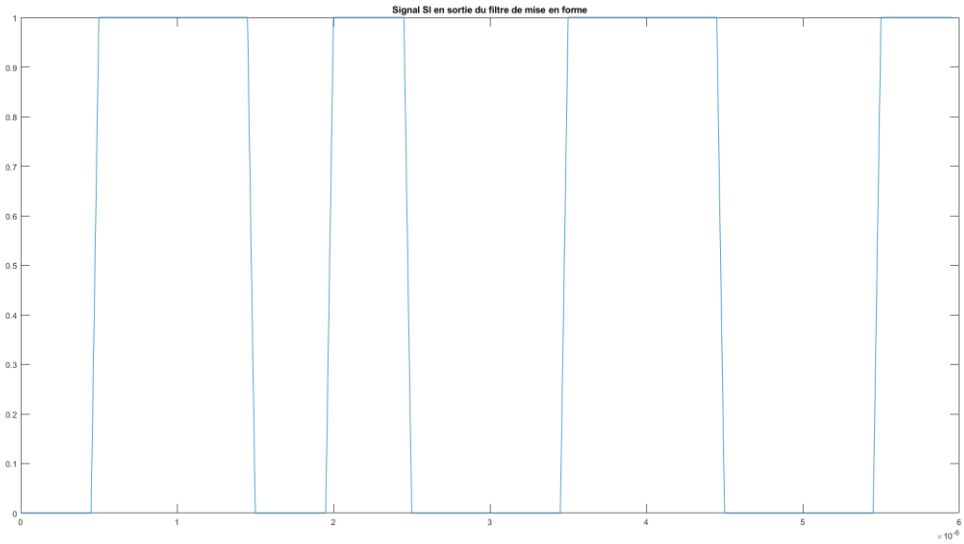


Figure 2: Signal $S_l(t)$ en sortie du filtre de mise en forme

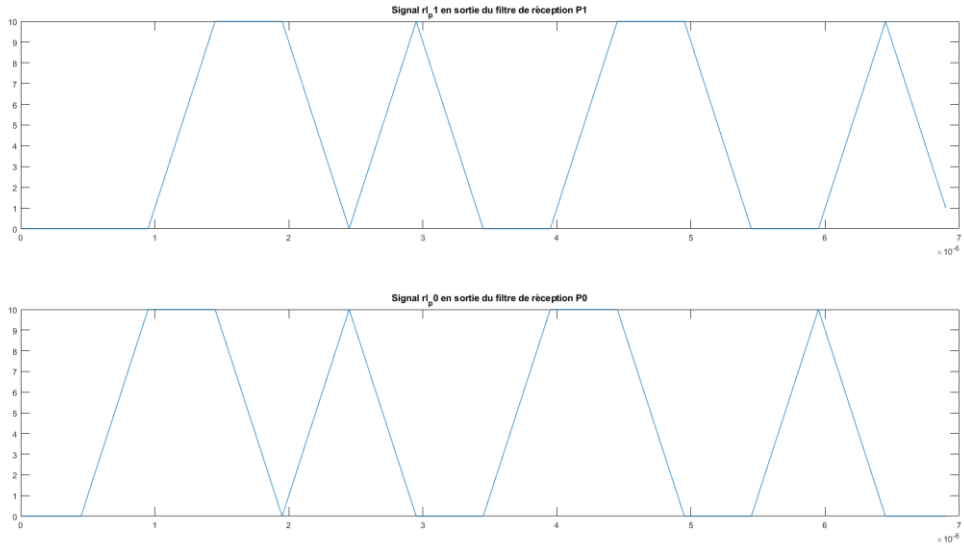


Figure 3 : Signaux $rl_{p1}(t)$ et $rl_{p0}(t)$ respectivement en sortie des blocs $p_1^*(-t)$et $p_0^*(-t)$

Question 2 : En comparant graphiquement les graphiques de $r_{0(k)}$ et $r_{1(k)}$, on en déduit que nous devons commencer l'échantillonnage à t = Ts, pour obtenir le rapport signal sur bruit le plus fort lors de la comparaison de ces deux signaux dans le bloc « arg max ».

Question 3 : A partir de la figure 2 on peut déduire l'expression de $s_l(t)$ sous la forme suivante :

$$s_l(t)0.5 + \sum_{k \in Z} A_k p(t - kTs)$$

Avec $A_k = \begin{cases} 1, & si\ b_k = 0 \\ -1, & si\ b_k = 1 \end{cases}$

Et p(t) est la forme d'onde biphase donnée dans *la figure 4* ci-dessous.

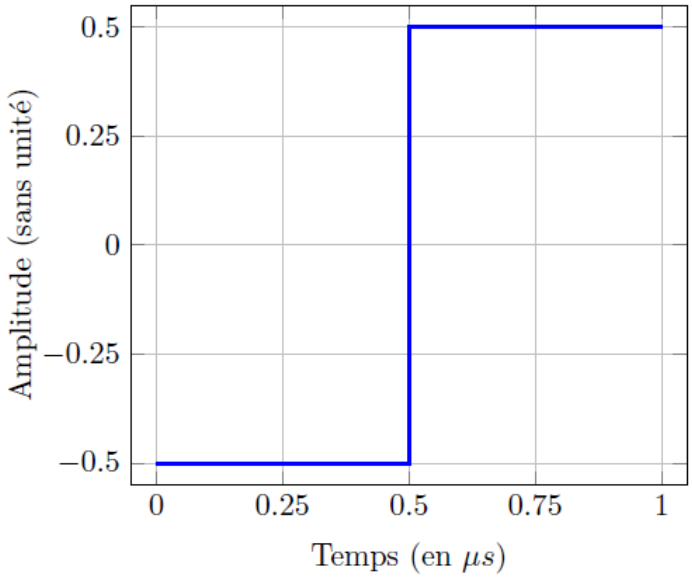


Figure 4 : Filtre de mise en forme "biphase" p(t) (code manchester). Le filtre est nul en dehors de l'intervalle $[0,1\mu s]$

La modulation obtenue est appelée une modulation d'amplitude en code Manchester.

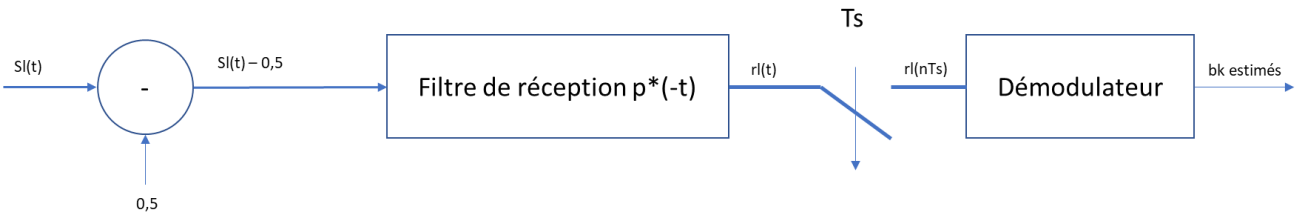


Figure 5 : Nouvelle chaine de communication

Question 4 : Les filtres $p_0^*(-t)$ et $p_1^*(-t)$ représentent les filtres de réceptions. Ce sont des filtres adaptés aux deux précédents filtres $p_{0(t)}$ et $p_{1(t)}$. Des filtres dit « adaptés » ont l'avantage d'apporter un meilleur rapport signal sur bruit. En effet, ils permettent de compenser le retard induit par $p_{0(t)}$ et $p_{1(t)}$.

Question 5 : Pour savoir si les couples de filtres $\left(p_0(t), p_0^*(-t)\right)$ *et* $\left(p_1(t), p_1^*(-t)\right)$ vérifient le critère de Nyquist, il faut vérifier que : $p_0(t) * p_0^*(-t) = \delta(\mathrm{n})\mathrm{v}_0[0]$ et $p_1(t) * p_1^*(-t) = \delta(\mathrm{n})\mathrm{v}_1[0]$.

Pour les deux couples, nous obtenons le même graphique (figure 6).

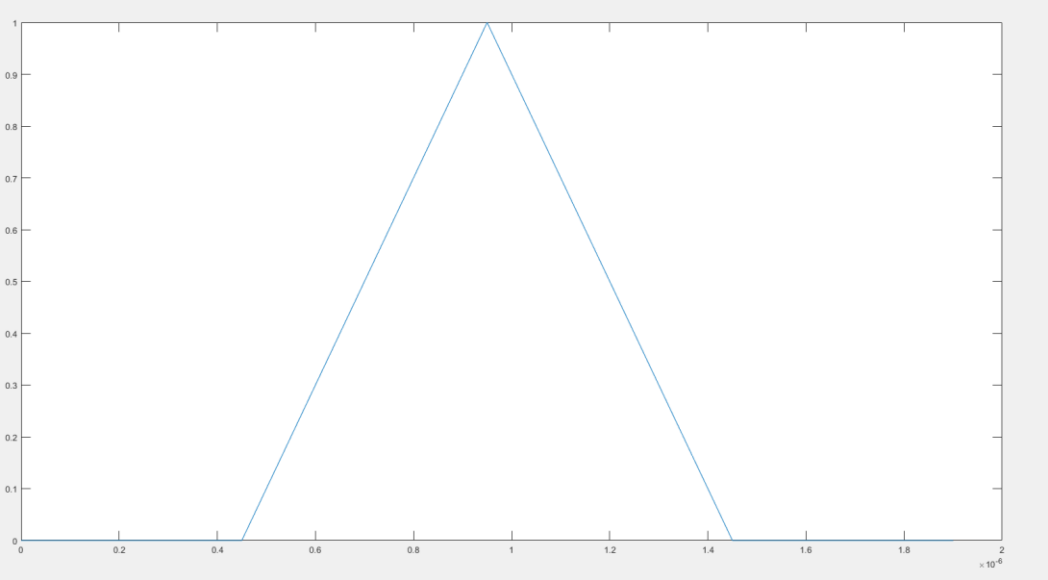


Figure 6 : Convolution de $\mathrm{p_0(t)} * \mathrm{p_0^*(-t)}$ *et* $\mathrm{p_1(t), p_1^*(-t)}$

Echantillonné en Ts on obtient bien l'équivalent d'un dirac d'amplitude $v_{0[0]}\ ou\ v_{1[1]}$. Le critère de Nyquist est respecté.

Question 6 : Calculons le moment d'ordre 1 du signal $s_{l(t)}$ noté $m_{s_l}(t)$:

$$\begin{aligned} m_{s_l}(t) &= \mathrm{E}[\mathrm{s_{l(t)}}] \\ &= E[0.5] + p(t - kTs) * E[A_k] \\ &= 0.5 + p(t - kTs) * \left(\frac{1}{2} * 1 - \frac{1}{2} * 1\right) \\ &= 0.5 \end{aligned}$$

On constate que le moment d'ordre 1 du signal $s_{l(t)}$, ne dépend pas du temps t. Ainsi on peut noter $m_{s_l}(t) = m_{s_l}$.

Question 7 : Remarque **:** Le signal étant réel $s_l^*(t+\tau) = s_l(t+\tau)$.

$$R_{s_l}(t,\tau) = E[s_l(t)s_l^*(t+\tau)]$$

$$= E\left[\left(0.5 + \sum_{k \in Z} A_k p(t - kTs)\right)\left(0.5 + \sum_{k' \in Z} A'_k p(t + \tau - k'Ts)\right)\right]$$

$$= E[0.25 + \sum_k \sum_{k'} A_k A'_k p(t - kTs)p(t - \tau - k'Ts) + 0.5 \sum_{k' \in Z} A_k p(t - kTs) + 0.5 \sum_{k' \in Z} A_k{}' p(t + \tau - k'Ts)]$$

$$= E[0.25 + \sum_k \sum_{k'} A_k A'_k p(t - kTs)p(t - \tau - k'Ts), \qquad car\ E[A_k] = 0$$

$$= 0.25 + \sum_{k}\sum_{k'} E[A_k A'_k]p(t - kTs)p(t - \tau - k'Ts)$$

Pour $\boldsymbol{k} \neq \boldsymbol{k}$ *on a* $E[A_k A'_k] = E[A_k]E[A'_k] = 0$, donc $R_{s_l}(t,\tau) = 0.25$

Pour $\boldsymbol{k} = \boldsymbol{k}'$ *on a* $E[A_k^2] = 1 * \frac{1}{2} + 1 * \frac{1}{2} = 1 = Ak(\sigma^2)$ *la variance des Ak, ainsi :*

$$R_{s_l}(t,\tau) = 0.25 + \sum_{k} \sigma^2\, p(t + \tau - kTs) + p(t - kTs)$$

$$R_{s_l}(t,\tau) = 0.25 + \sum_{k} p(t + \tau - kTs) + p(t - kTs)$$

Question 8 : On cherche à montrer que $s_l(t)$ est cyclo-stationnaire, de période de cyclo stationnarité Ts. On cherche donc à montrer mathématiquement que $m_{s_l}(t) = m_{s_l}$ (voir question 6) et que $R_{sl}(t + Ts, \tau) = R_{sl}(t,\tau)$.

$$R_{s_l}(t + Ts, \tau) = E\left[\left(0.5 + \sum_{k\in Z} A_k p(t + Ts - kTs)\right)\left(0.5 + \sum_{k'\in Z} A_k{}' p(t + Ts + \tau - k'Ts)\right)\right]$$

On pose $k = k - 1$ *et* $k' = k' - 1$, et on obtient

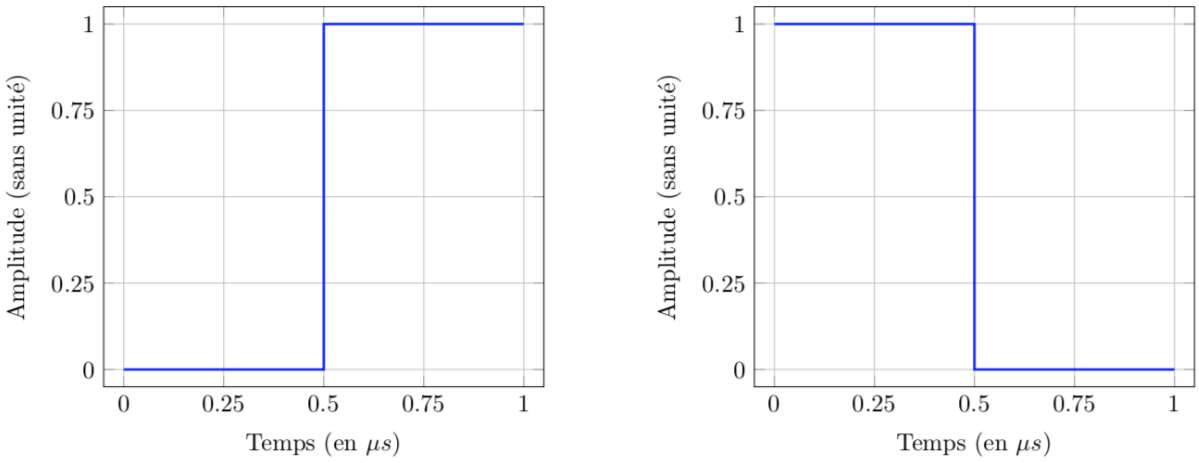
$$E\left[\left(0.5 + \sum_{k\in Z} A_k p(t - kTs)\right)\left(0.5 + \sum_{k'\in Z} A_k{}' p(t + \tau - k'Ts)\right)\right] = R_{sl}(t,\tau)$$

On a démontré que $R_{sl}(t + Ts, \tau) = R_{sl}(t,\tau)$. Donc $s_l(t)$ est cyclo-stationnaire de période de cyclo stationnarité Ts.

III- Implémentation de la chaîne de communication

b. La modulation PPM (Pulse Position Modulation)

En entrée de la chaîne de communication, un signal binaire appelé b_k est reçu. Il va rentrer dans le bloc PPM. Ce bloc PPM va encoder les informations binaires reçues en un signal appelé s_l grâce aux filtres de mise en forme $p_0(t)$ et $p_1(t)$. Ces filtres-là sont respectivement représentés en figure 2(a) et figure 2(b).



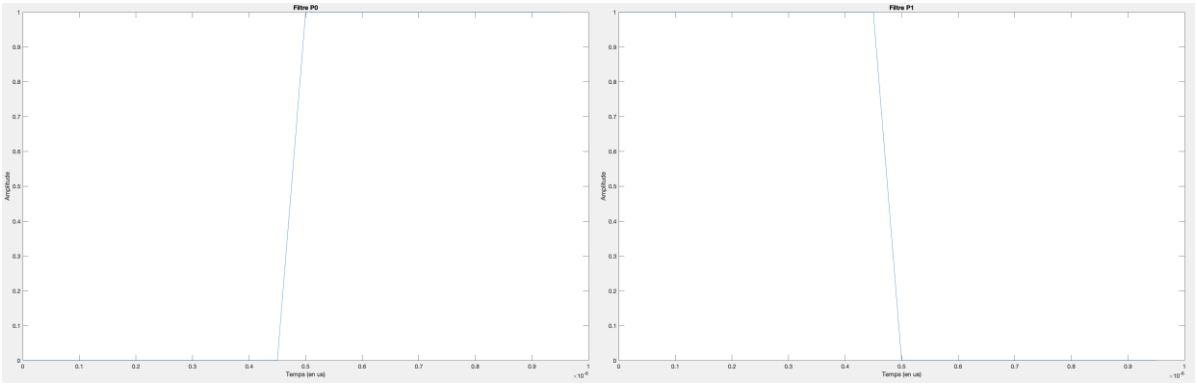
(a) Impulsion $p_0(t)$ encodant le bit 0 (b) Impulsion encodant $p_1(t)$ le bit 1

Figure 7. Impulsions de base pour la modulation par position

Ces deux filtres ont été implémentés sous Matlab en utilisant les fonctions « ones » et « zeros » et le facteur de sur-échantillonnage Fse comme suit :

```
p1 = [ones(1, Fse/2), zeros(1, Fse/2)]; % Filtre p0
p0 = [zeros(1, Fse/2), ones(1, Fse/2)]; % Filtre p1
```

Après avoir tracé les courbes des filtres de mise en forme pour vérifier notre code, nous avons obtenu les filtres représentés en figure 3(a) et figure 3(b).



Les filtres n'ont pas un temps de montée aussi faible que ceux exposés en figure 2. En effet, le facteur de sur-échantillonnage permet d'avoir 20 échantillons par microseconde, ce qui rend le temps de montée réel un peu moins performant que le temps de montée théorique. Le signal en sortie de ce premier bloc est s_I et a pour expression :

$$s_I(t) = \Sigma\, p_{b_k}(t - kT_s)$$

sachant que T_S représente ici le temps du filtre de mise en forme ($p_0(t)$ ou $p_1(t)$) et que

$$p_{b_k}(t) = \begin{cases} p_0(t), \text{ si } b_k = 0 \\ p_1(t), \text{ si } b_k = 1 \end{cases}$$

Pour le signal binaire suivant : « 011010 », on obtiendra le signal $s_I(t)$ représenté sur la figure 4, en ayant ajouté bout à bout les signaux des filtres de mise en forme en fonction du bit échantillonné comme suit :

```
for i=1:1:Nb
    if bits(i) == 1
        sl=[sl,p1];
    else
        sl=[sl,p0];
    end
end
```

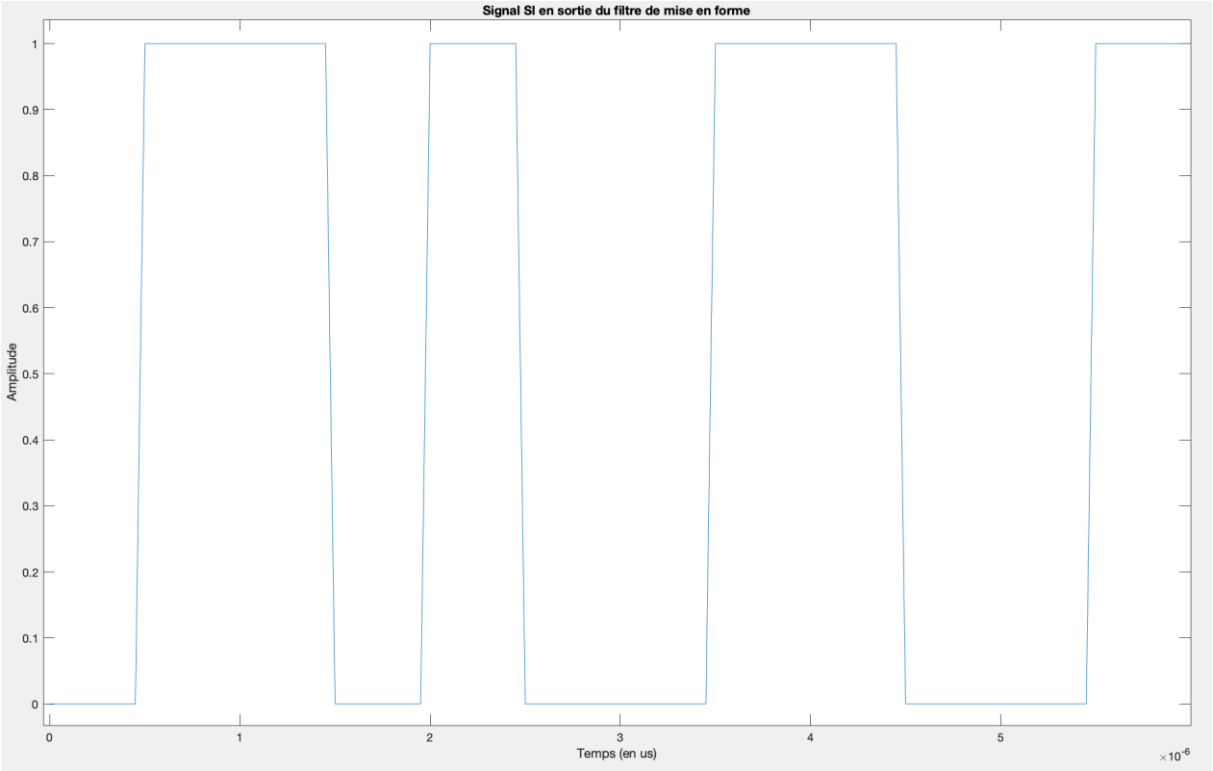


Figure 8. Signal s_l en sortie du filtre de mise en forme

c. Le canal de propagation

Le canal de propagation sera considéré idéal et sera modélisé par l'expression suivante :

$$h(t) = \delta(t)$$

Cela induit que le signal en sortie de l'émetteur sera celui reçu à l'entrée du récepteur.

d. Les filtres de réception $p_0^*(-t)$ et $p_1^*(-t)$

Les filtres $p_0(t)$ et $p_1(t)$ ont pour filtre conjugué $p_0^*(t)$ et $p_1^*(t)$. Ces filtres ont exactement les mêmes caractéristiques que $p_0(t)$ et $p_1(t)$ car ceux sont des filtres réels sans partie imaginaire. La composante temporelle négative permet simplement de préparer le filtre à la convolution avec le signal arrivant du canal de propagation.

Les deux filtres ayant pour seule différence un décalage temporel, les signaux en sortie des deux filtres de réception auront la même enveloppe mais décalée dans le temps. Le code Matlab pour réaliser cette convolution est le suivant :

```
rl_p0 = conv(sl, fliplr(p0)); % convolution par p0*(-t) filtre adapté de p0
rl_p1 = conv(sl, fliplr(p1)); % convolution par p1*(-t) filtre adapté de p1
```

En traçant les deux signaux reçus en sortie des deux filtres de réception, nous obtenons les courbes visibles à la figure 5.

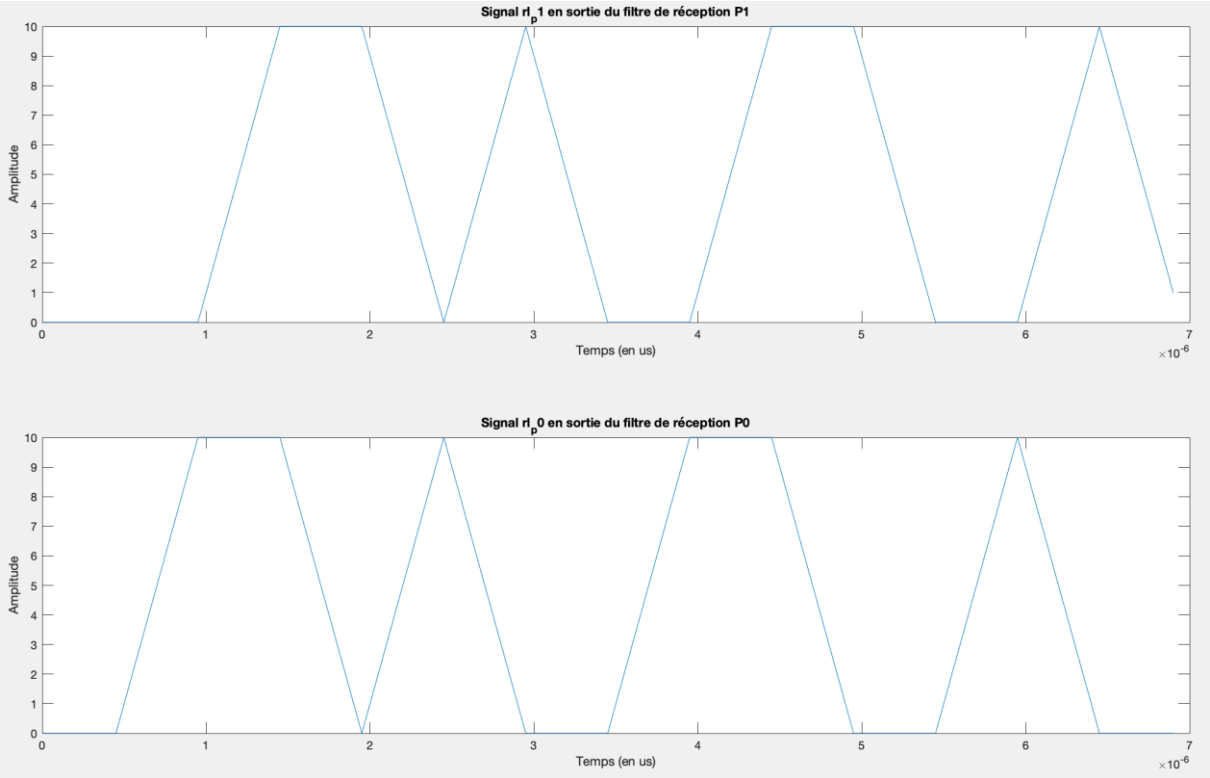


Figure 9. Signaux en sortie des filtres de reception $p_0^*(t)$ et $p_1^*(t)$

Nous voyons à travers cette figure que les signaux ont un décalage temporel de 1µs, correspondant à T_s. Le bloc suivant dans la chaîne de communication est le bloc Arg max. Ce bloc permet de comparer les deux signaux échantillonnés au rythme T_s et d'en faire une comparaison. En effet, si à $t=kT_s$, $r0(k) \geq r1(k)$ alors bk =0, sinon bk = 1.

Il faut donc trouver le meilleur moment auquel commencer l'échantillonnage des deux signaux afin d'avoir le meilleur rapport de comparaison possible, i.e. d'avoir toujours un signal à un niveau logique de 0 quand l'autre signal a un niveau logique de 1.

Pour cela, nous avons décidé d'échantillonner les deux signaux à partir de T_s comme le montre le code Matlab suivant :

```
rln_p0 = rl_p0(Fse:Fse:end); % échantillonnage au ryhtme Ts de rl_p0
rln_p1 = rl_p1(Fse:Fse:end); % échantillonnage au ryhtme Ts de rl_p1
```

Cela nous permet d'avoir deux signaux échantillonnés comme représentés sur la figure 6.

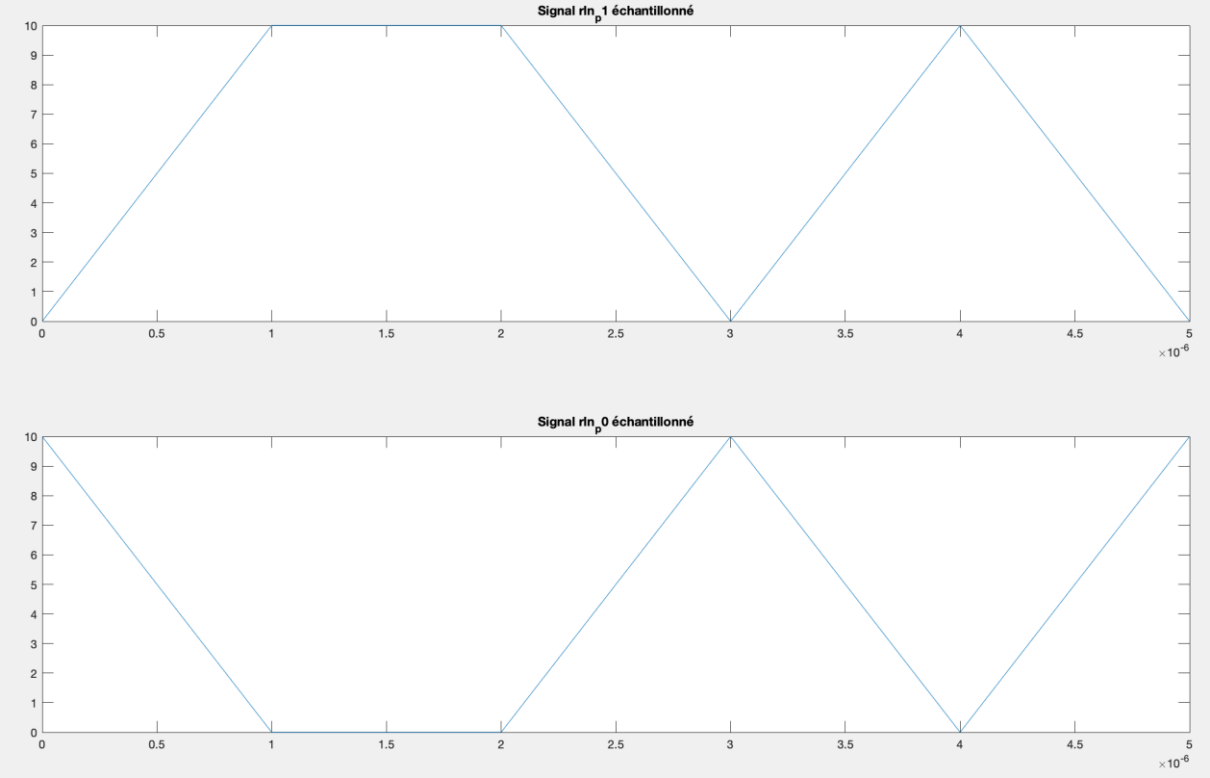


Figure 10. Signaux en sortie des filtres de réception échantillonnés au rythme T_s à partir de T_s

A travers cette figure, nous pouvons remarquer que les deux signaux sont exactement opposés l'un à l'autre, ce qui nous permettra d'avoir une très bonne comparaison entre les deux signaux pour faire l'estimation des bits reçus.

e. Estimation des bits reçus

Une fois que les deux signaux en sortie des filtres de réception sont échantillonnés, ils arrivent en entrée du bloc Arg max qui va effectuer une comparaison entre les deux signaux afin d'estimer la valeur des bits reçus. Le code Matlab pour réaliser ce bloc est le suivant :

```
for i=1:1:Nb
    if rln_p1(i) <= rln_p0(i)
        bk(i) = 0;
    else
        bk(i) = 1;
    end
end
```

Ce code permet de venir comparer chaque échantillon des deux signaux afin de déterminer s'il s'agit d'un bit à 1 ou à 0. Un contrôle avec un BER (Bit Error Rate) permet de vérifier si l'implémentation du bloc Arg max engendre une erreur ou non :

```
BER = mean(abs(bits - bk)) % Taux d'erreur binaire
```

La variable « bits » contient le flux binaire envoyé et la variable « bk » contient les bits estimés en sortie du bloc Arg max. Après avoir simulé toute la chaîne de communication, un BER de 0 est obtenu, ce qui confirme bien que les bits estimés sont exactement les mêmes que les bits envoyés en entrée.

f. Estimation des bits en présence de bruit

Pour montrer que le bruit généré par l'environnement autour de la chaîne de communication est dérangeant pour l'estimation des bits, un bruit aléatoire a été créé sous Matlab :

```
for i=1:11
   n0(i) = Eb/(10.^(Ebn0(i)/10));   % n0 en fonction de (n0/Eb)
   Variance(i) = n0(i)/2;           % variance
   nl = (randn(1,length(xl))*sqrt(Variance(i))); % génération du bruit en
fonction de la variance
   yl = xl + nl; % signal + bruit

  (passage par les filtre de mise en forme + échantillonnage + estimation des bits)

   BER(i) = mean(abs(bits - bk)) % Taux d'erreur binaire
```

Pour des valeurs de E_b/N_0 comprises entre 0dB et 10dB, un bruit aléatoire a été généré en fonction de la variance de N_0. Les différentes BER associés aux valeurs de E_b/N_0 ont été enregistrés dans un vecteur afin de tracer la courbe du BER en fonction du rapport du signal / bruit et de mettre en évidence que plus le rapport signal / bruit est élevé, plus le BER sera faible. Cette remarque est représentée en figure 7.

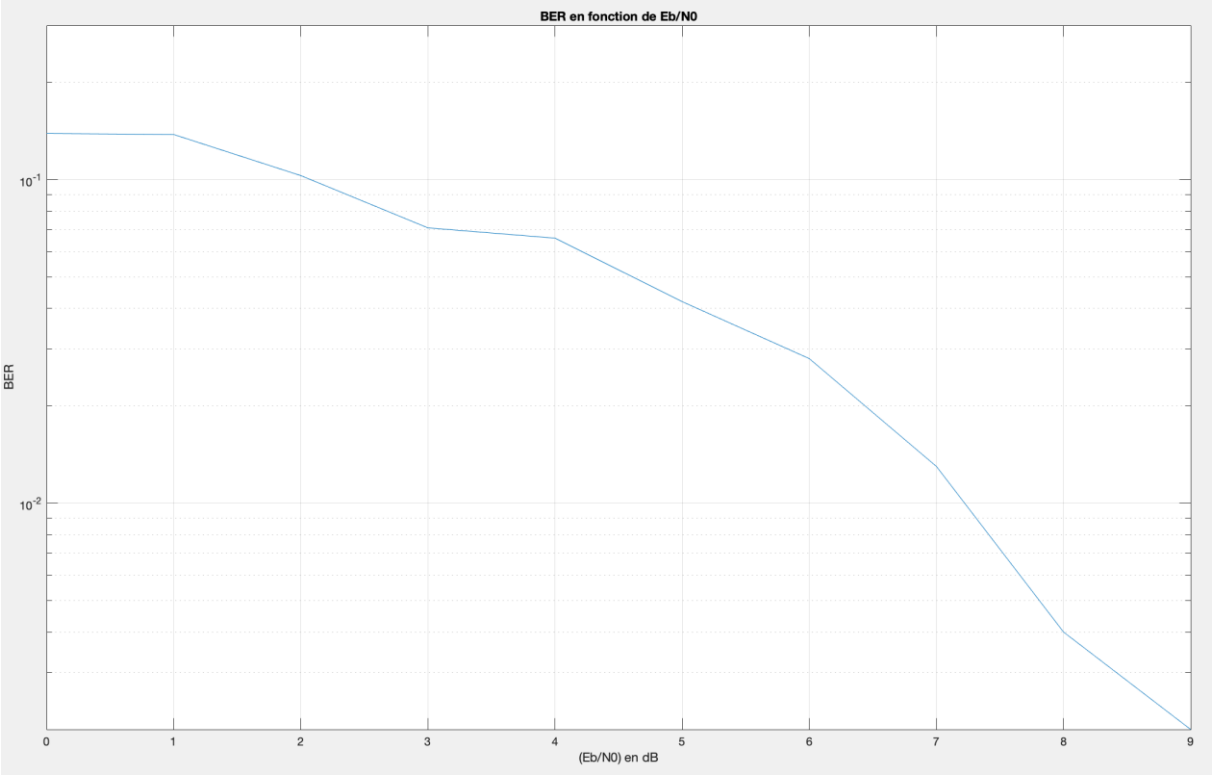


Figure 11. BER en fonction du rapport signal/bruit E_b/N_0

Je ne sais pas quoi dire sur la DSP

IV- Synchronisation en temps et en fréquence

Lors du développement de la chaîne de communication sous Matlab, nous n'avons pas pris en compte deux défauts qui peuvent subvenir :

- La synchronisation temporelle : le signal subit un délai de propagation δ_t, il faut le compenser ;
- La synchronisation fréquentielle : l'effet Doppler introduit par le mouvement de l'avion ainsi que les défauts d'oscillateurs locaux introduisent un décalage en fréquentiel δ_f. Ces deux effets sont pris en compte avec le modèle en bande de base suivant :

$$y_l(t) = s_l(t - \delta_t)e^{-j2\pi\delta_f t} + n_l(t),$$

avec $n_l(t)$ le bruit généré, δ_t et δ_f représentent respectivement les désynchronisations temporelle et fréquentielle du signal.

a. Synchronisation en fréquence

La synchronisation en fréquence est la plus simple à réaliser. En effet, le signal reçu est seulement multiplié par $e^{-j2\pi\delta_f t}$. En calculant la valeur absolue du signal reçu, la composante fréquentielle disparaîtra car le module d'un exponentiel est 1. Il ne restera donc que le signal utile voulu. Le code Matlab suivant permet de réaliser la synchronisation fréquentielle :

```
xl_M = abs(xl); % calcul du module
```

b. Synchronisation en temps

La synchronisation est réalisée à la réception en utilisant un signal $s_p(t)$ de durée T_p = 8µs appelé préambule et envoyé en entête des trames ADS-B. Le préambule est le signal donné en Figure 8.

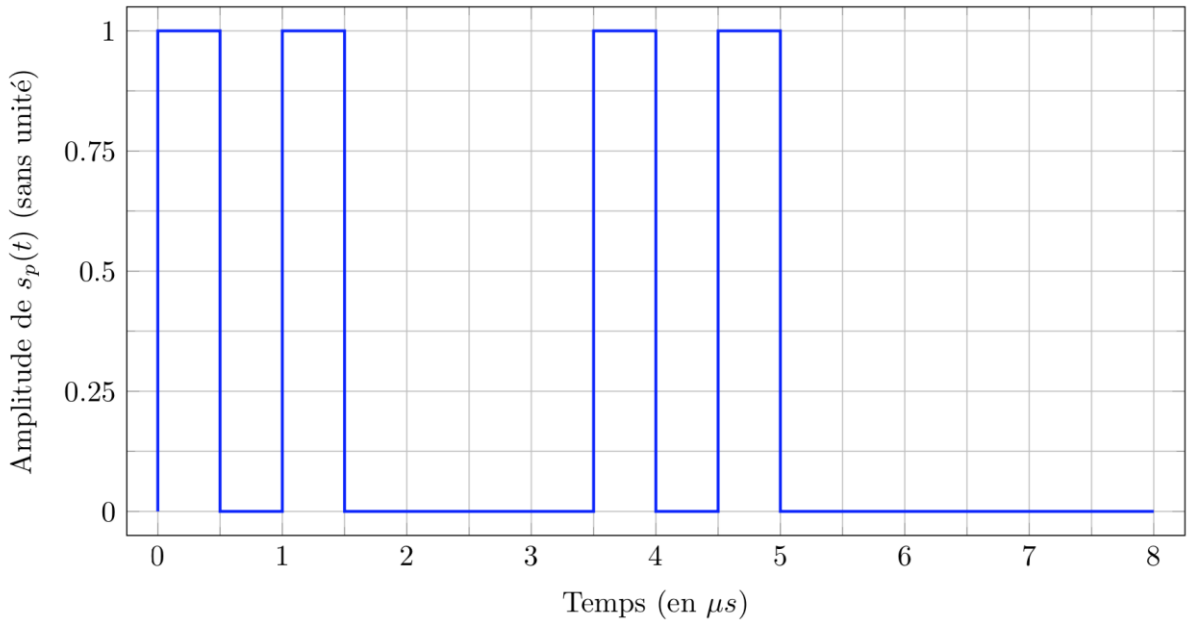


Figure 12. Préambule de détection pour la synchronisation temporelle

Pour pouvoir détecter ce préambule dans les trames ADS-B, il faut réaliser

V- Traitement et décodage de signaux réels

c. Structure des trames ADS-B

Les signaux émis par les appareils pour l'ADS-B ont une durée de 120 s. Ils sont constitués des parties suivantes :

— le préambule (identique à celui de la section précédente),
— le format de la voie descendante,
— la capacité,
— l'adresse OACI (Organisation de l'Aviation Civile Internationale) de l'appareil,
— les données ADS-B,
— les bits de contrôle de parité.

La durée de chacune des parties ainsi que leur position dans une trame ADS-B sont représentées en Figure 8. Afin d'éviter des redondances inutiles avec les explications issues de l'énoncé, nous détaillerons uniquement les parties utiles à la suite du projet.

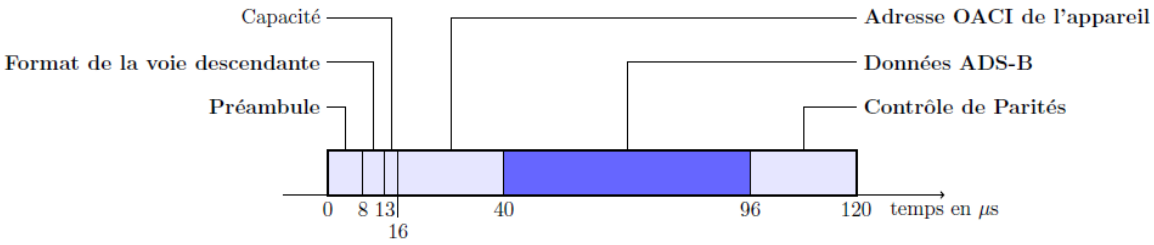


Figure 13 : Structure d'une trame ADS-B

Dans notre cas, les données utiles pour récupérer des informations de positions en vol et d'identification de l'appareil se situent uniquement dans la partie « Données ADS-B ».

Les 5 premier bits des Données ADS-B constituent le FTC pour Format Type Code et permettent de connaitre le type de données qui suit.

Question 18
Si cette valeur convertie en décimale notée $FTC = [b_1, b_2, b_3, b_4, b_5]$, avec b_1 le bit de poids fort est comprise entre 1 et 4 tout deux inclus, alors nous auront à faire à une trame contenant un message d'identification.
Si cette valeur est comprise entre 9 et 22 tout deux inclus, alors le message correspondra à un message de position en vol.

Codage (question 19 et 20)

La fonction bit2registre prend en argument un vecteur colonne de 112 bits et un registre à mettre à jour. Elle extrait les informations du vecteur binaire et renvoie le registre mis à jour seulement si le CRC ne détecte pas d'erreur.

Dans un premier temps nous nous assurons que le registre de sortie recopie le registre d'entrée :

```
%% Initialisation (on recopie le vecteur d'entrée en sortie)
registre_out = registre_in;
```

Ensuite nous testons le CRC, nous effectuons le test à l'aide de fonctions fournies.

```
% Test sur CRC
CRC_detector = crc.detector('0xFFF409'); % polynôme du checksum
[trame_utile, crcErrFlag] = detect(CRC_detector, data(:)); % retourne les
données utiles(trame_utile) ainsi qu'un flag d'erreur(crcErrFlag) sur le
checksum
% crcErrFlag = 1 : erreur
% crcErrFlag = 0 : RAS
```

Nous effectuons la suite du programme uniquement si `crcErrFlag == 0` (aucune erreur détectée). Dans le cas contraire, le programme aura uniquement recopié le registre d'entré en sortie (partie Initialisation).

Nous isolons la trame ADSB utile, et nous la mettons sous la forme d'un vecteur ligne, car nous avions déjà coder la suite du décryptage en se basant sur des vecteurs lignes.

```
    % Isolation de la trame ADSB en un vecteur ligne
    donnee_utile_ligne = trame_utile'; % permet de faire la transposer pour
récupérer un vecteur ligne
```

```
    trame_ADSB = donnee_utile_ligne(33:88); % on isole la trame des données
ADSB
```

Nous calculons à présent le FTC pour savoir à quel type de trame nous avons à faire (voir question 19). Si le FTC est compris entre 9 et 22 inclus tous les deux, alors nous décodons une trame transportant un message de position en vol, sinon si celui-ci est compris entre 1 et 5, tous deux compris nous décodons une trame contenant un message d'identification.

Pour décoder les trames de message d'identification nous avons créé la fonction `ConvertCarIdentification(bits_car)` qui prend en argument 6 bits issus d'une trame d'identification correspondant à un caractère comme référé à la figure 13. Et qui renvoi le caractère décodé associé en fonction du tableau de caractère de la figure 14.

Remarque : nous considérons SP comme un espace.

Index binaire	Champs
1 : 5	MSB … LSB — Format Type Code
6 : 8	MSB … LSB — Catégorie de l'appareil
9 : 14	MSB … LSB — Caractère 1 de l'identifiant
15 : 20	MSB … LSB — Caractère 2 de l'identifiant
21 : 26	MSB … LSB — Caractère 3 de l'identifiant
27 : 32	MSB … LSB — Caractère 4 de l'identifiant
33 : 38	MSB … LSB — Caractère 5 de l'identifiant
39 : 44	MSB … LSB — Caractère 6 de l'identifiant
45 : 50	MSB … LSB — Caractère 7 de l'identifiant
51 : 56	MSB … LSB — Caractère 8 de l'identifiant

Figure 14 : Composition du message d'identification

				b_6	0	0	1	1
				b_5	0	1	0	1
b_4	b_3	b_2	b_1					
0	0	0	0			P	SP	0
0	0	0	1		A	Q		1
0	0	1	0		B	R		2
0	0	1	1		C	S		3
0	1	0	0		D	T		4
0	1	0	1		E	U		5
0	1	1	0		F	V		6
0	1	1	1		G	W		7
1	0	0	0		H	X		8
1	0	0	1		I	Y		9
1	0	1	0		J	Z		
1	0	1	1		K			
1	1	0	0		L			
1	1	0	1		M			
1	1	0	0		N			
1	1	1	1		O			

Figure 15 : Tableau de caractères

Enfin si nous avons à faire à une trame contenant un message de position en vol, nous utilisons le tableau ci-dessous pour isoler les informations à décoder.

Index binaire	Champs
1 ⋮ 5	MSB Format Type Code LSB
6 7	MSB Surveillance Status LSB
8	Indicateur de type d'antenne
9 ⋮ 20	MSB Altitude LSB
21	Indicateur de temps UTC
22	Indicateur de format CPR
23 ⋮ 39	MSB Latitude encodée avec CPR LSB
40 ⋮ 56	MSB Longitude encodée avec CPR LSB

Figure 16: Composition du message de position en vol

Les parties qui nous intéressent et que nous décoderons sont les parties « Format Type Code », « Altitude », « Indicateur de format CPR », « Latitude encodée avec CPR », « Longitude encodée avec CPR ».

```
    if (Ftc >= 9) && (Ftc <= 22) % Ftc correspondant à Message de position
en vol
        altitude_bits = trame_ADSB(9:20); % On extrait les bits
correspondant aux différents blocs d'informations
        format_CPR = trame_ADSB(22);
        latitude_CPR = trame_ADSB(23:39);
        longitude_CPR = trame_ADSB(40:56);
```

Les calculs effectués sur les vecteurs ainsi extraits ne seront pas détaillés ici afin d'éviter un rapport trop long ainsi qu'une redondance avec l'énoncé. Le code commenté fournit en annexe s'auto suffit. Nous nous sommes contenté d'implémenter dans Matlab les formules fournies dans l'énoncé.

Finalement, le code une fois fini nous le testons à partir du fichier `trames_20141120.mat` fourni et contenant plusieurs trames ADS-B issues d'un seul avion.

En utilisant la fonction « plot » de matlab nous affichons la trajectoire ainsi que le nom de l'avion qui ont était décodés à partir de trames du fichier `trames_20141120.mat`. Le résultat est présent ci-dessous en figure 16.

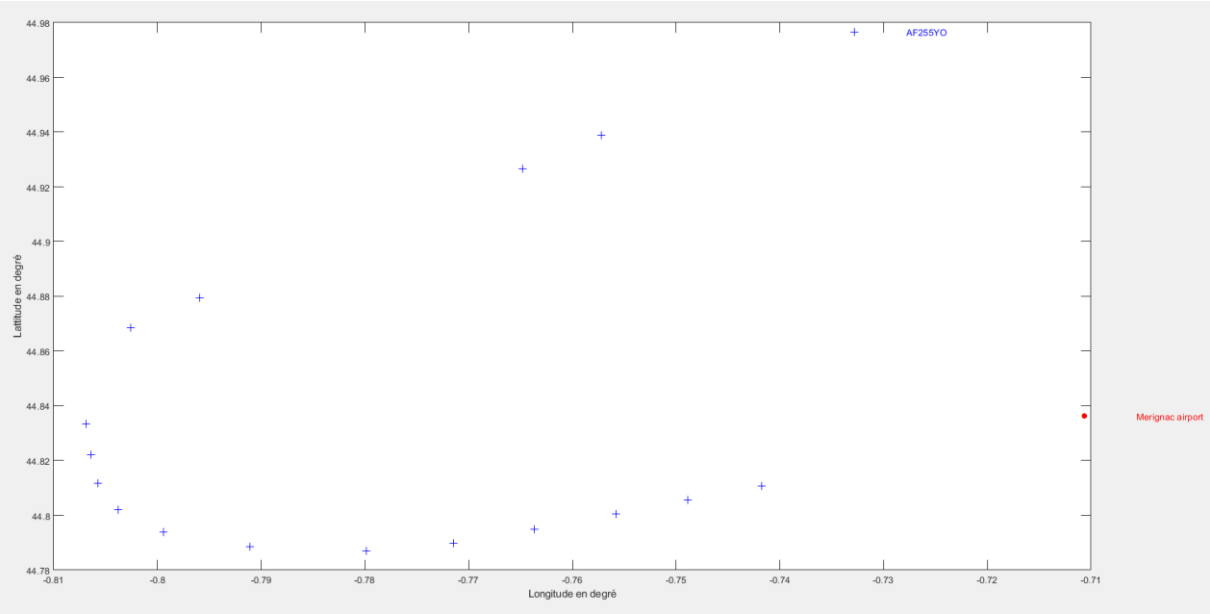


Figure 17 : Trajectoire d'un avion issue des trames ADS-B fournies

Interprétation des résultats + question 21

Les données décodées à partir des trames fournies contenues dans `trames_20141120.mat`, nous indique un vol avec comme Identifiant AF255Y0. En se reportant sur le site https://fr.flightaware.com/live/flight/AFR255, nous apprenons que c'est un Boeing 777-300ER de la compagnie AirFrance qui se dirigeait vers Paris.

Si nous observons la trajectoire courbée ainsi que les altitudes croissantes à chaque position (en les affichants dans matlab), nous pouvons en déduire que nous avons à faire à un décollage. Sur notre plot, le point de départ semble être l'aéroport de Mérignac.