# **Suivi du travail effectué – Stage assistant ingénieur**

# École Royale Militaire

## Semaine 1 : (du Mardi 2 Juin 2020 au Dimanche 7 Juin)

Je me suis familiarisé avec le sujet. Celui-ci, rappelé par mon tuteur sous la forme suivante :

*« Étudier la contre-mesure radar active, donc un émetteur "trompant" un autre radar. On suppose dans un premier temps (pour simplifier) qu'aussi bien le radar que la cible se trouvent dans un plan horizontal. L'idée est que la cible possède deux (ou plus) émetteurs (dont elle peut éventuellement contrôler l'espacement (ex: sur un petit flotteur ou une petite plateforme (UGV/UAV) télécommandée) cohérents et qu'elle les utilise pour produire les échos d'une fausse cible qui est interprétée comme provenant d'une autre direction. »*

Après avoir lu le chapitre correspondant à la formation de faisceaux, et m’être replongé dans mon cours de Traitement du signal Avancé (UE 4.2), j’ai retrouvé différentes méthodes permettant de faire du beamforming. CBF, MUSIC, et MVDR, chacun ayant ses avantages et inconvénients.

J’ai donc commencé à réfléchir à la modélisation du problème. N’ayant pas le matériel adéquat, j’ai opté pour simuler le tout sur un python notebook et éventuellement d’autres fichiers python pour alléger le code. Ce Notebook fait office de démonstrateur : y sont expliqués les principes physiques, avec des images, mais aussi et surtout les différentes courbes obtenues.

En utilisant un modèle de mes rapports de l’école, et en y apposant le logo de l’ERM, j’ai maintenant un modèle Word pour débuter mon rapport de stage. Bien entendu, il faudra que je me renseigne sur les attentes vis-à-vis de ce rapport.

J’ai aussi mis en place un Git sur mon répertoire de travail, relié à un dépôt Github.

<https://github.com/MaxENSTAB/Stage2A>

J’y ai plusieurs branches : une pour chaque grande étape de mon modèle.  
🡪 Sur la branche master, j’ai commencé à modéliser le beamforming sur une cible immobile

🡪 Sur mouvement, je ferai bouger ma cible

🡪 Sur emission, j’analyserai le signal reçu et émettrai un signal brouillant les relevés des RADAR.

La difficulté de cette semaine aura été de réfléchir aux données du problème afin qu’elles reflètent un minimum la réalité, mais aussi de modéliser correctement les algorithmes, parfois difficile à comprendre.

Au final, j’ai un notebook qui affiche en fonction des directions estimées de la position des sources, quel que soit le nombre. Sachant qu’on sait où sont les sources avant les estimations, il est possible d’indiquer s’il y a faux-positif, ou non-détection de source.

## Semaine 2 : (du Lundi 8 Juin au Dimanche 14 Juin)

Après avoir obtenu des résultats prometteurs, j’ai voulu me pencher sur la mise en mouvement de ma cible. J’ai donc créé une fonction qui me donne un certain nombre de s-uplets, représentant les angles et les amplitudes de chaque signal source sur ma cible. Je modifie l’angle et l’amplitude pour coller à un mouvement prédéfini.

Ce mouvement sera uniforme, dans une direction colinéaire à notre ULA, avec une vitesse constante. Étant donné que la cible effectue un certain nombre de mesures pour afficher un résultat (30 mesures), il faudrait afficher le dit résultat toutes les secondes, ou demi secondes, et animer le tout pour se rendre compte du mouvement.

Beaucoup de temps passé sur matplotlib, cette librairie n’étant pas forcément adaptée à Jupyter, il m’a fallu apprendre d’autres méthodes d’affichage pour pouvoir animer mes courbes.

Git : création de la branche, et adaptation du code pour coller aux nouvelles variables et fonctionnalités. J’ai quand même eu des soucis, étant donné que je suis utilisateur débutant, mais tout fonctionne normalement.

J’ai aussi commencé ce journal de bord, afin de garder une trace écrite de mes journées de travail, et servir de base à mon rapport. Le log de mon Git permet aussi de retracer toutes les actions effectuées. Grâce aux commentaires de chaque commit, je peux retracer ma progression et mes problèmes.

J’ai démarré à proprement parler la partie émission, où je fais de nouvelles hypothèses de départ, afin d’avoir un temps de calcul réduit.

En relisant les premiers échanges de mail avec le Pr. Neyt, je me suis rappelé qu’il avait mentionné deux plateformes sur lesquelles implémenter le banc : HackRF, et N310 de NI. J’ai déjà utilisé du matériel de National Instruments, mais de manière vraiment simplifiée (un accéléromètre).

## Semaine 3 : (du Lundi 15 Juin au Dimanche 21 Juin)

J’ai envoyé le travail effectué au Pr. Neyt, lequel m’a conforté dans l’idée de faire ce notebook. Cependant, il y avait quelques rectifications à faire : les signaux reçus par la cible doivent être cohérents.

C’est pourquoi les lignes suivantes ont été remplacées, pour générer des signaux carrés cohérents.

temp = rd.randn(S,L)

*##################################################################*

n = sig\_n\*(rd.randn(M,L)) *#Gaussien centré*

s = np.sin(2\*np.pi\*c0\*temp/4) *#Je génère un signal aléatoire*

ÉNORMÉMENT D’HEURES DE TRAVAIL POUR TROUVER LES SIGNAUX CARRÉS

j’ai changé le BeamformingMethods.py en conséquence, et ai renommé BeamformingNotebook en DoANotebook.ipynb

Dans un second notebook noté BeamEmissionNotebook.ipynb, j’ai explicité le calcul permettant d’émettre un faisceau. Dans une boucle, j’ai fait en sorte « d’émettre » dans la direction de réception d’un écho RADAR. Pour émettre à partir d’un ULA, on j’utilise un beamformer simple, se basant sur une différence de phase entre les émetteurs.

J’ai mis en place la situation décrite par le Pr. Neyt dans son mail, à savoir : « L'idée maintenant est de considérer deux cibles dont les échos sont corrélés. Donc ils proviennent de deux directions différentes (les steering vectors sont différents), mais l'amplitude fluctue de la même façon. Et l'idée est de voir ce qui se passe si l'on écarte les cibles l'une de l'autre (on commence par deux cibles confondues---et dans ce cas-là, la position sera correcte). »