|  |  |
| --- | --- |
| Une image contenant texte, carte  Description générée automatiquement | |
| **Stage Assistant Ingénieur**  **-**  **Ingénieur des Études et Techniques de l’Armement Maxime BARRET** | |
| Une image contenant chemise  Description générée automatiquement | École Royale Militaire  Rue Hobbema, 8  1000 Bruxelles, Belgique  BARRET, Maxime, maxime.barret@ensta-bretagne.org |

# Remerciements

Je tiens à remercier Mr Ali Kenchaf, de m’avoir orienté vers l’École Royale Militaire pour ce stage. Malgré le contexte actuel, j’en retiens de bonnes expériences qui ont permis de développer mon esprit d’initiative.

Merci au Pr. Xavier Neyt de m’avoir donné ce sujet, et avoir encadré mon stage d’assistant ingénieur.

De plus, je tiens à remercier Élie et François pour le soutien, la solidarité et l’esprit d’entraide dont ils ont su faire preuve durant ce stage.

Enfin je remercie ma compagne, Julia, de m’avoir soutenu dans mes initiatives et pour sa patience sans faille.

# Résumé

Mon stage d’assistant ingénieur s’est entièrement déroulé en ligne, et sur ordinateur. Axé sur la formation de faisceaux et la simulation, mon travail a été réalisé sur l’outil Jupyter Notebook[[1]](#endnote-1).

Dans ce rapport, nous considérerons que nous sommes aux commandes d’un drone aérien ou terrestre dont la discrétion est primordiale. La position et la vitesse de ce drone peuvent être déterminées par un ou plusieurs RADAR « ennemis » : le but est ici d’utiliser la formation de faisceaux pour tromper ces RADAR, et ainsi transmettre des informations contradictoires ou effacer la présence du drone. Après détermination du domaine d’application des différentes méthodes de formation de faisceaux, plusieurs simulations sur Jupyter ont été faites pour correspondre à des situations réalistes, et analyser les résultats.

# Abstract

My internship as a deputy engineer was solely remote, on my computer. Based on beamforming and simulation, the results of my work were coded on Jupyter Notebooks.

In this report, we will consider ourselves to be in charge of a UAV or UGV of which stealth is primordial. The drone’s position and speed can be determined by enemy RADAR systems : our goal is to make use of beamforming in order to send contradictory information or even make the drone disappear. After determining the validity domain of the many beamforming methods available, several Jupyter simulations were conducted to be as close as possible to reality and draw conclusions.

Sommaire

[Remerciements 2](#_Toc52311602)

[Résumé 3](#_Toc52311603)

[Abstract 3](#_Toc52311604)

[Introduction 6](#_Toc52311605)

[I. Contexte académique et sanitaire 7](#_Toc52311606)

[1. École Royale Militaire 7](#_Toc52311607)

[2. Département d’accueil 8](#_Toc52311608)

[3. Déroulement du stage 8](#_Toc52311609)

[4. Actions prises et outils utilisés 8](#_Toc52311610)

[II. Objectifs du stage 10](#_Toc52311611)

[1. Comprendre la situation décrite 10](#_Toc52311612)

[2. Processus de contre-mesure 10](#_Toc52311613)

[3. Attentes du stage 10](#_Toc52311614)

[4. Solutions atteignables 11](#_Toc52311615)

[III. Théorie – Beamforming 12](#_Toc52311616)

[1. Introduction 12](#_Toc52311617)

[2. Mise en équation 13](#_Toc52311618)

[3. *Directions of Arrival* (DoA) 15](#_Toc52311619)

[3.1. *Classical Beamforming* 15](#_Toc52311620)

[3.2. *MVDR* 15](#_Toc52311621)

[3.3. *MUSIC* 15](#_Toc52311622)

[4. Émission de l’écho trompeur 17](#_Toc52311623)

[IV. Simulation – Les notebooks 17](#_Toc52311624)

[1. Présentation de l’outil 17](#_Toc52311625)

[2. DoANotebook 17](#_Toc52311626)

[3. DoAmouvement 20](#_Toc52311627)

[4. Simulation 23](#_Toc52311628)

[Conclusion 23](#_Toc52311629)

[Table des figures 23](#_Toc52311630)

[Annexes 24](#_Toc52311631)

[Annexe 1 – Suivi du travail effectué 24](#_Toc52311632)

[Annexe 2 – Notebooks et fichier python 27](#_Toc52311633)

[Annexe 3 – Ressources Matlab pour le notebook Simulation 27](#_Toc52311634)

[Glossaire technique 29](#_Toc52311635)

# Introduction

Dans l’industrie de défense, les ingénieurs civils ou militaires sont amenés à travailler sur des systèmes complexes ayant des capacités toujours plus à la pointe de la technologie : cela permet de s’adapter aux différents contextes opérationnels pouvant être rencontrés sur les différents théâtres.

Dans un contexte opérationnel, il est nécessaire de pouvoir s’affranchir des capacités d’un appareil ennemi. Dans notre cas, nous souhaiterions être capable de tromper un RADAR ennemi. Des solutions passives permettent de réduire la *RADAR Cross Section* [[2]](#endnote-2) ou même d’absorber une grande partie des ondes reçues, mais il est aussi possible d’utiliser des solutions actives. Émettre un signal dont la fréquence, la phase et l’amplitude sont générées de façon à imiter un écho RADAR provenant d’un autre endroit permettrait d’être tout aussi furtif. Afin de pouvoir émettre le signal dans une direction intéressante vis-à-vis de la situation, il est préférable de disposer d’outils à forte directivité. Cette directivité doit aussi s’adapter au contexte électromagnétique en présence, et aux positions des sources éclairant notre système. Les *Uniform Linear Arrays[[3]](#endnote-3)* (ULA) remplissent ces prérequis.

Le Professeur Xavier Neyt m’a proposé le sujet suivant :

« **Conception d’un banc de contre-mesure RADAR – Production d’un faux écho par Beamforming** »

Le but de ce stage a été de confronter les compétences acquises en cours d’année aux activités du département d’accueil, ainsi que de se familiariser avec les techniques de formation de faisceaux avec les *ULA*, les algorithmes de détection de directions d’arrivée, dans le but de créer un banc de contre-mesure RADAR.

Ce rapport est indissociable des notebooks présents en annexes, et du site GitHub. En effet, tous les notebooks présents possèdent des animations, qu’il est nécessaire de visionner pour comprendre ce qui est écrit.

# Contexte académique et sanitaire

## École Royale Militaire

Située en plein cœur de Bruxelles, l’École Royale Militaire est un établissement militaire d’enseignement universitaire, chargé de la formation militaire, académique et physique des officiers de toutes les composantes belges. Fondée par le roi Leopold 1er en 1834 afin de fournir à la Belgique des officiers professionnels et bien formés, dont les qualités sont à la fois athlétiques et intellectuelles. L’École Royale Militaire dispense une formation pluridisciplinaire répondant aux exigences de la défense et de la société.

Ce modèle de formation, semblable en de nombreux point à la formation des Officiers des Corps de l’Armement[[4]](#endnote-4), m’a tout de suite attiré afin d’obtenir de nouvelles compétences et aptitudes dans les domaines qui me seront d’une grande utilité dans mon parcours professionnel.

Une image contenant extérieur, bâtiment, grand, rue

Description générée automatiquement

Figure : École Royale Militaire de Belgique

L’établissement possède aussi un centre de recherche dans les domaines vastes de la défense et de la sécurité. Ce centre permet de développer des systèmes technologiquement avancés à partir des différentes recherches des départements le composant.

## Département d’accueil

Le département m’ayant accueilli durant l’été est le CISS : *Communication, Information, Systems & Sensors*.

Mon tuteur, le Prof. Dr. Ir.[[5]](#footnote-1) Xavier Neyt, fait partie de ce département, et dirige l’unité de recherche sur le traitement d’image.

Les recherches et les travaux d’ingénierie effectués dans ce département se concentrent sur l’acquisition, le traitement, et l’interprétation de données multimodales dans le but d’assister une prise de décision.

Répondant au besoin constant de montée en niveau technologique, ce département est particulièrement sollicité de par l’évolution du contexte opérationnel. Les équipements et infrastructures dont le département bénéficie sont multiples et variés, tout comme les sujets traités par les unités de recherche : une chambre semi-anechoïque, une plateforme de réception RADAR bistatique, un cluster de calcul, et bien d’autres.

## Déroulement du stage

Dès le mois de Septembre, je me suis mis à chercher un stage d’assistant ingénieur correspondant à mes appétences. Pour cela, l’expériences de mes prédécesseurs était importante à prendre en compte. En discutant avec plusieurs d’entre eux, j’ai été orienté vers le Pr. Ali Kenchaf. Ce dernier m’a alors proposé de prendre contact avec le Pr. Neyt pour un stage à l’École Royale Militaire.

Le début du stage était initialement prévu le 5 Mai, cependant la COVID-19 a bouleversé les conventions établies, et nous a obligé à faire preuve d’inventivité et de résilience afin que le stage se déroule sans encombre. L’établissement n’était pas disposé à nous accueillir début Mai, et n’ayant pas l’autorisation de sortir du territoire à ce moment-ci, mes camarades[[6]](#footnote-2) et moi-même avons préféré consolider nos connaissances académiques. Avec l’accord de notre tuteur, nous avons pu suivre la seconde partie du semestre 4 de l’ENSTA Bretagne à distance.

Cette partie terminée, nous avons recontacté notre tuteur début Juin afin de commencer à réfléchir à nos sujets respectifs, dont les intitulés nous ont été donnés par mail. Nous devions faire une réunion en visio-conférence, mais les indisponibilités récurrentes du Pr. Neyt ne nous ont pas permis de nommer les objectifs, ni d’avoir une base de notations et de termes en commun, développant ainsi notre esprit d’initiative et notre autonomie.

## Actions prises et outils utilisés

Le télétravail s’étant imposé comme la solution la moins contraignante pour mener à bien ce stage en faisant fi des difficultés sanitaires, il était nécessaire d’utiliser des outils en ligne collaboratifs. En effet, travailler sur des fichiers stockés localement augmenterait considérablement la quantité d’échanges de mails, et aurait été une perte de temps.



Figure - Logo de Jupyter, outil utilisé durant le stage

Après un succinct échange avec le Pr. Xavier Neyt signifiant le début de mon stage, j’ai pris l’initiative de simuler les situations sur des notebooks, et de mettre en place un *repository*[[7]](#endnote-5) GitHub®[[8]](#footnote-3) afin de garder une trace de mon travail, mais aussi faciliter les éventuelles interventions de mon tuteur sur le code déjà réalisé.

Afin de rester joignable en tout temps et en tout lieu, j’ai utilisé Skype Entreprise®, logiciel préconisé par mon tuteur, mais aussi Discord® pour pouvoir discuter avec mes camarades sur des questions techniques. Ces outils sont tout particulièrement adaptés au contexte du télétravail.

# Objectifs du stage

## Compétences à acquérir

Après avoir vu le sujet proposé par le Pr. Neyt, il était nécessaire de faire un état des lieux des compétences utiles voire indispensable pour mener à bien ce stage. Comme tout stage d’assistant ingénieur, mais aussi dans une grande majorité de postes d’ingénieurs diplômés, les acquis d’école d’ingénieurs ne sont qu’un tremplin vers d’autres sujets plus complexes, éloignés ou non de ce qui a déjà été appris.

Il ne faut pas ignorer ce qui a été appris, mais plutôt le maitriser pour passer au niveau supérieur. Les mots-clés qui m’ont interpelé dans l’intitulé du sujet sont « RADAR », et « *beamforming* ». Si le RADAR est une notion qui m’est familière, de par les cours reçus au Semestre 4, le *beamforming* l’est moins. La lecture de l’intitulé du sujet s’est immédiatement suivie d’une relecture des cours d’Ondes, Environnement et Systèmes, ainsi que de recherches sur la formation de faisceaux.

Il s’avère que la formation de faisceaux est aussi utilisée pour des signaux acoustiques, avec un réseau de microphones : une introduction au *beamforming* nous a été dispensée lors du Semestre 4. Il m’a alors suffit de me renseigner sur les différences induites par le passage aux signaux RADAR, pour mieux comprendre les enjeux du sujet.

## Comprendre la situation décrite

Afin de m’expliquer au mieux quelles étaient ses attentes vis-à-vis de moi, le Pr. Xavier Neyt m’a proposé une mise en situation. Un drone terrestre, aérien, ou maritime, se déplace dans un plan, et se fait éclairer[[9]](#footnote-4) par un ou plusieurs RADAR hostiles. Cette situation n’est pas souhaitable, car un RADAR peut dévoiler notre position, notre vitesse et certaines propriétés intrinsèque de notre drone. Ces informations peuvent mettre à mal la discrétion d’une mission dans un contexte opérationnel, mais aussi alimenter les autodirecteurs d’engins anti-drones.

L’idée pour contrer l’acquisition de données par le camp adverse sera de falsifier les données reçues par le RADAR. Pour cela il faut envoyer un signal perturbateur : ce dernier peut augmenter le RSB[[10]](#endnote-6), ou interférer de manière destructive avec la réflexion du signal RADAR.

Pour illustrer ses propos, je me suis proposé de modéliser cette situation en utilisant le langage Python[[11]](#footnote-5).

## Processus de contre-mesure

Le principe de contre-mesure que l’on souhaite mettre en place a été expliqué précédemment, il faut toutefois détailler le procédé, et le décortiquer en plusieurs phases.

* Dans un premier temps, nous allons d’abord estimer les directions d’arrivés des signaux RADAR hostiles. Il existe plusieurs algorithmes d’estimation de *DoA[[12]](#endnote-7)*
* Dans un second temps, ces signaux vont être analysés dans le but de savoir quels signaux envoyer pour brouiller les pistes.
* Enfin, un écho trompeur sera émis par notre drone, dans les directions d’arrivées

Dans cette situation, nous utiliserons des *ULA*. Contrairement à une simple antenne omnidirectionnelle, un ULA va nous fournir les outils d’estimation des *DoA*, mais aussi la directivité souhaitée pour émettre l’écho trompeur vers les RADAR. Cet écho pourra être un bruit puissant pouvant saturer son récepteur, ou un signal donnant de fausses pistes.

## Attentes du stage

Compte-tenu de la situation et de l’objectif décrit, une contre-mesure sera efficace si chaque étape se déroule correctement. Nous simulerons une situation simple, avec plusieurs RADAR et une source, afin de nous familiariser avec les diverses notions et notations à adopter. Aussi, étant donné que toutes les notions sont nouvelles pour moi, il est nécessaire de poser un cadre simple pour commencer, et ne pas s’éparpiller à vouloir faire une simulation complète d’un bout à l’autre.

L’objectif du stage était toujours de concevoir un banc, et donc d’utiliser des équipements pour générer et recevoir des signaux : deux solutions étaient proposées par le Pr. Neyt. Tout d’abord, une initiative open source nommée HackRF[[13]](#footnote-6) est un périphérique permettant de recevoir ou transmettre des émissions RF[[14]](#footnote-7) allant de 1MHz à 6 GHz. Une autre possibilité était celle d’utiliser la plateforme propriétaire de National Instruments®, cette plateforme étant à la fois matérielle et logicielle.

En attendant les directives précisant que le stage ne se fera pas en présentiel, je me suis intéressé uniquement de manière superficielle à ces solutions. J’ai déjà eu affaire à la plateforme de National Instruments® pour utiliser un accéléromètre, système bien plus simple à mettre en place. Ici, il faut utiliser des réseaux de capteurs, l’acquisition et le traitement de données sont plus compliqués à coder avec le logiciel propriétaire[[15]](#footnote-8) de National Instruments®.

## Solutions atteignables

Le contexte dans lequel ce stage a été effectué n’a pas permis de manipuler du matériel, quel qu’il soit. De ce fait, seules des simulations ont été à ma portée. Cela permettra de développer plusieurs compétences essentielles : parfaire mes connaissances en Python, mais aussi utiliser de manière plus poussée les outils de visualisation, et d’algèbre linéaire. De plus, l’utilisation quotidienne de l’outil GitHub par le terminal, puis par l’outil graphique, est un incontournable du travail collaboratif en ligne.

Les simulations seront détaillées en IV. Tout au long de ce rapport et des fichiers en annexes, les problèmes sont considérés plans, les bruits gaussiens additifs, et les ondes en champs lointain.

Une image contenant intérieur, ordinateur, assis, petit

Description générée automatiquement

Figure - Opérateur RADAR sur la frégate Chevalier Paul en 2014 - Marine Nationale®

# Théorie – Beamforming

## Introduction

Le *Beamforming*, ou formation de faisceaux, est une technique de traitement du signal utilisée avec des réseaux de capteurs afin d’émettre ou de recevoir des signaux directionnels. Dans le cas où le réseau d’antenne est utilisé pour recevoir un signal, nous l’appellerons *beamformee*, et *beamformer* dans le cas contraire.

Les signaux peuvent être sonores, mais aussi électromagnétiques, ce qui est le cas pour nous.

Pour connaitre la direction dans laquelle on reçoit une onde incidente dans une direction principale, le principe de base est le suivant : le déphasage induit par la différence de chemin entre chaque capteur.

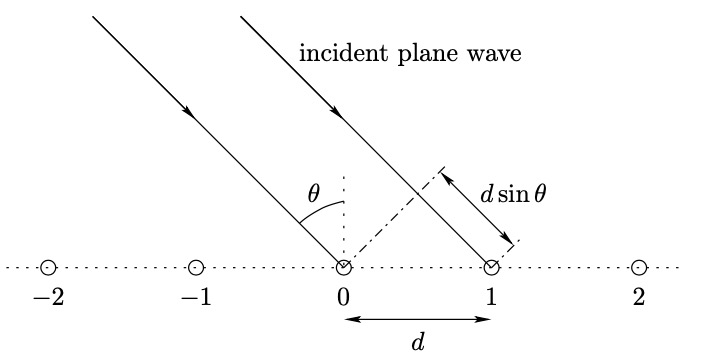


Figure - Différence de chemin entre deux capteurs successifs - 'Signaux et Sytèmes', Prof. Dr Ir Xavier Neyt

Ce déphasage permet d’estimer l’angle entre la normale aux capteurs, et la direction de l’onde incidente.

[1]

Pour ce qui est de l’émission, il suffit d’induire un déphasage à chaque capteur : on favorise donc l’émission dans une émission particulière, comme décrit à la figure 5.

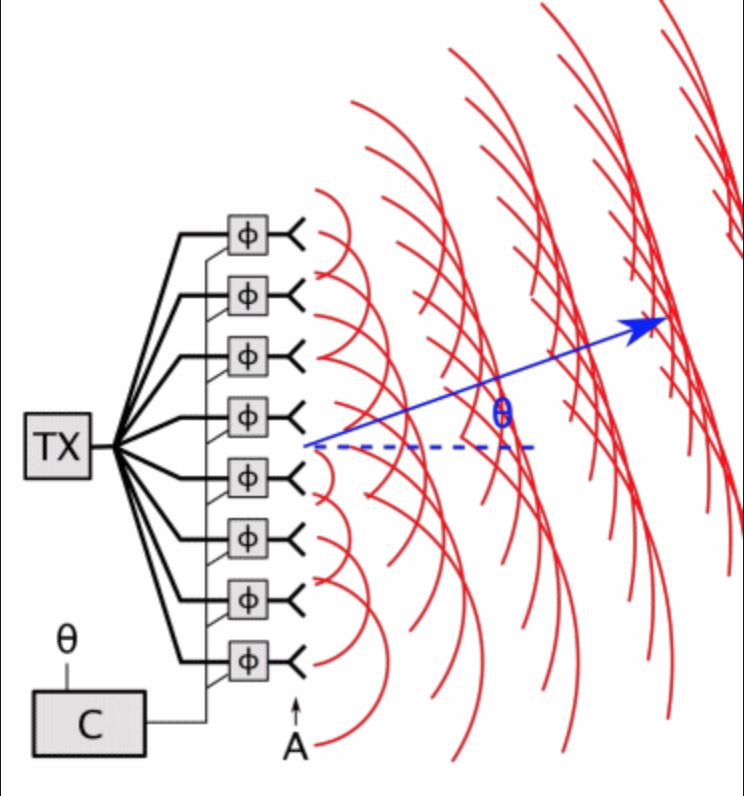


Figure - Illustration de la directivité des ULA

## Mise en équation

Dans cette partie seront données les équations régissant nos simulations, ainsi que les objets mathématiques clés pour comprendre les notions en jeu.

Afin de définir notre *ULA*, il est nécessaire de donner une distance *d*, un nombre de capteurs *M*. La grandeur correspond au rapport entre la fréquence de l’onde (émise ou reçue) et de la célérité de la lumière dans le vide.

Notre *ULA* va effectuer plusieurs *snapshot[[16]](#footnote-9)* selon l’équation suivante :

Où

* S est le nombre de sources RADAR éclairant notre cible
* correspond à l’onde envoyée par la s-ième source RADAR
* est la direction dans laquelle se trouve la s-ième source
* est le *steering vector[[17]](#endnote-8) correspondant à la direction*
* est un bruit additif

Le *steering vector* est défini par le déphasage induit entre deux capteurs successifs, de la manière suivante :

Ce vecteur va simuler le déphasage que subirait une onde provenant de la direction .

Enfin, à partir de notre *snapshot*, nous allons établir une matrice de covariance de la manière suivante :

Nous verrons que cette matrice de covariance est primordiale pour les algorithmes de *DoA.*

Il y a aussi des précautions à prendre quant à la résolution spatiale !

Au même titre que la transformée de Fourier nécessite une certaine résolution fréquentielle, le *beamforming* doit respecter certaines règles de bases, dont les équivalences sont rappelées ci-dessous.

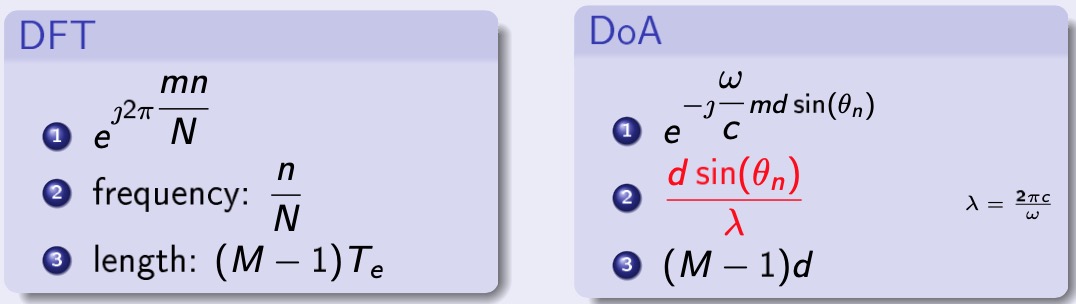


Figure - Équivalence des conditions Transformée de Fourier / DoA : Sources localization with sensor array, C. Vanwynsberghe - A. Drémeau

Avec ces équivalences, le critère de Shannon-Nyquist s’adapte à notre cas, et se met sous la forme suivante :

Nous devons donc faire attention à l’espacement entre nos capteurs, compte-tenu de la classe d’onde à laquelle on s’attend.   
  
Enfin, il faut au moins autant de capteurs que de sources à détecter.

## *Directions of Arrival* (DoA)

Lorsqu’éclairé par un ou plusieurs RADAR, et pour mettre en place notre stratégie de contre-mesure, il est nécessaire de connaître avec une certaine précision la ou les directions d’arrivée des signaux RADAR. Plusieurs méthodes existent pour les *ULA*, chacune avec leurs avantages et leurs inconvénients. Notre but est ici de les énumérer : les avantages et inconvénients de chaque méthode seront détaillés dans la partie IV. [1]

Chaque méthode nous donne une manière de calculer la puissance reçue par notre *ULA* : à nous d’interpréter les courbes formées et d’en déduire les directions, donnée par les maxima.

### *Classical Beamforming*

Cette méthode, comme son nom l’indique, est la plus simple et la plus intuitive à mettre en place.

Nous allons calculer la puissance reçue par notre *ULA* pour chaque angle selon la formule suivante :

Les maxima de cette fonction, sur une grille allant de nous permet d’identifier les directions d’intérêt.

### *MVDR*

MVDR, ou *Minimum Variance Distortionless Response* ou encore CAPON, est une méthode de beamforming permettant de minimiser la variance du signal reçu, et d’ainsi avoir une détection plus précise. La puissance est donnée par :

On notera tout de même que pour calculer cette puissance, il faut d’abord calculer l’inverse d’une matrice. est de taille (M,M), ce qui correspond au nombre de capteurs. Si ce nombre s’était avéré trop grand, il aurait été impossible de faire tous les calculs assez rapidement.

### *MUSIC*

MUSIC est un nom mélodieux, signifiant *MUltiple Signal Classification*. Cette méthode ressemble à celle la précédente, mais est plus fine dans ce qu’elle analyse. En effet, MUSIC nous fait décomposer notre matrice de covariance en valeurs et vecteurs propres.

Les sous-espaces contenant le bruit et le signal sont donc séparés, de telle sorte que le sous-espace contenant le signal soit de dimension S [[18]](#footnote-10) et que les sous-espaces soient orthogonaux. Ainsi, aux directions d’intérêt, le sous-espace du bruit est nul.

Voici la fonction de puissance correspondante :

Où

* est la matrice correspondant au sous-espace du bruit

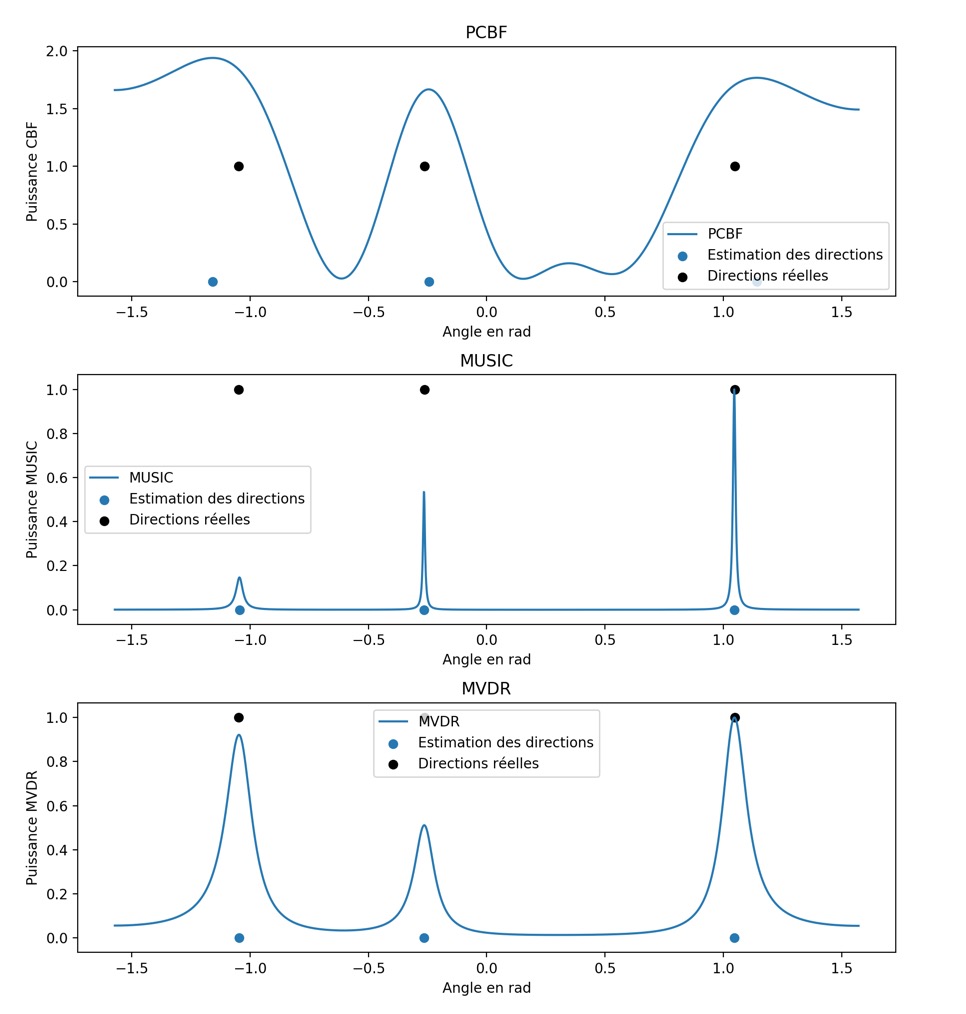


Figure - Illustration des différentes méthodes de DoA pour 3 sources

## Émission de l’écho trompeur

Après avoir détecté les directions d’arrivée, il faut donc y émettre un écho trompeur. Les émetteurs de notre *ULA* vont émettre un même signal **s**, mais déphasé, afin de générer un faisceau. C'est un *classic beamformer*.

Le signal envoyé, dans l’idéal, serait un signal de la même forme que celui reçu, tout en inversant sa phase : les interférences destructives avec l’écho renvoyé atténueront la visibilité de notre cible.

Le déphasage entre 2 émetteurs consécutifs sera de , comme décrit à la Figure 4. Dans les faits, ce déphasage est induit par des composants électronique en amont de l’*ULA*.

# Simulation – Les notebooks

## Présentation de l’outil

Un *notebook,* ou calepin, est un outil s’ouvrant dans un navigateur. Il est composé de cellules, qui peuvent être soit du texte (Titres, formules mathématiques, équations, images), ou du script exécutable (Python, Julia, R). Sur chacun de mes *notebook*, je détaille le titre, mes sources, et explique la situation. Ainsi, indépendamment les uns des autres, chaque notebook se suffit à lui-même pour les explications.

Afin d’alléger au maximum la lecture, j’ai fait en sorte de rassembler toutes les fonctions dans un fichier python à part, que j’importe au début. De même, afin de mieux illustrer le beamforming, j’ai utilisé la bibliothèque *Matplotlib*, et ai fait en sorte de pouvoir animer mes courbes.

Le début de chaque notebook sert à poser toutes les variables qui seront utiles tout au long, mais aussi à importer toutes les bibliothèques pouvant s’avérer utiles : *Matplotlib* pour l’affichage, *time* pour avoir le temps de calcul, *warnings* pour afficher des erreurs (faux-positif, détection manquée), mais aussi *Scipy* pour la génération de signaux et les outils d’analyse de courbes.

L’utilisation de notebook permet de modifier assez rapidement les valeurs introduites : ainsi, il est aisé de tester nos systèmes avec plus ou moins de capteurs, un bruit important ou non, ou encore des signaux émis différents.

## DoANotebook

Ce notebook est le premier que j’ai créé. Il m’a permis de poser les bases de ma simulation. On montre d’abord comment change le diagramme de rayonnement de notre *ULA* en changeant la direction d’émission principale. Cela nous permet dans un premier temps de nous habituer à l’utilisation de la bibliothèque *Matplotlib*, ainsi que de bien illustrer le fait que le gain est maximal pour une direction particulière.

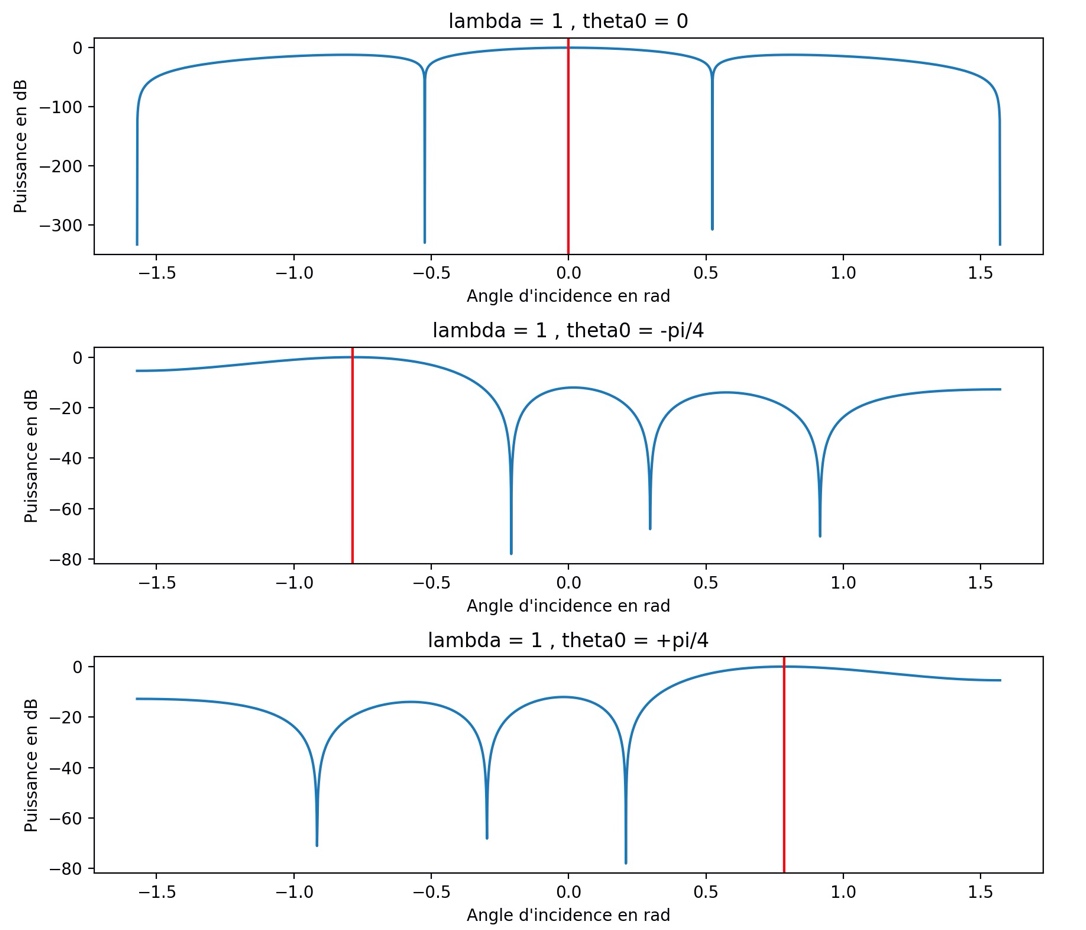


Figure - Diagrammes de rayonnement d'un ULA à 5 capteurs dans des directions données

Par la suite, j’ai construit plusieurs variables permettant de définir trois sources RADAR éclairant directement notre cible, que j’ai placé aux angles suivants : , , et . Toutes ces sources sont cohérentes, et émettent un signal carré de haute fréquence, auquel j’ai ajouté un bruit gaussien d’un certain écart-type.

Ces signaux, auxquels j’ai appliqué le *steering vector* pour simuler leurs directions respectives, et desquels j’ai déduit la matrice de covariance des *snapshots* reçus, nous fournissent toutes les données nécessaires pour appliquer les différentes méthodes de *DoA* détaillées dans la partie III. 3.

J’applique donc séparément les algorithmes CBF, MVDR et MUSIC, sur les données reçues par mon *ULA.* Les résultats sont stockés dans un dictionnaire, pour vérifier les données plus en détails. Une analyse des courbes permet de déterminer les pics des courbes de la puissance en fonction de l’angle d’incidence. Les abscisses des pics indiquent les directions estimées des sources : enfin, ces directions estimées sont comparées aux directions théoriques.

Afin de quantifier la qualité des algorithmes d’estimations, je vérifie dans un premier temps qu’il n’y a pas eu de faux-positifs ou de non-détections. Une de ces erreurs discrédite directement la méthode en question pour les paramètres sélectionnés. Si trois sources sont détectées, je vérifie alors l’erreur relative[[19]](#footnote-11) entre les directions estimées et théoriques, pour comparer plus précisément les méthodes.

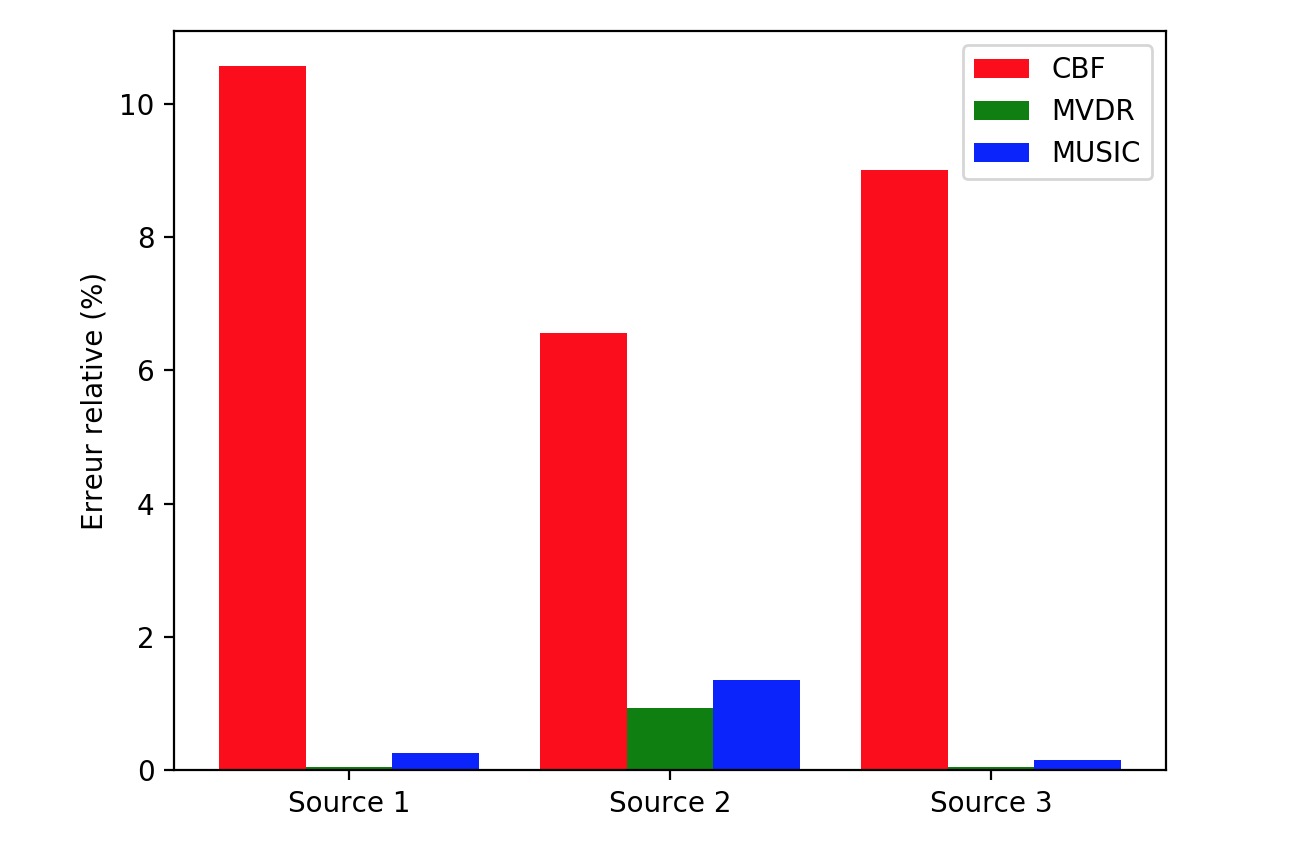


Figure - Comparaison des erreurs relatives sur les directions des sources, entre les trois méthodes décrites

Sans surprise, la méthode la plus simple et demandant le moins de calculs s’avère la moins précise pour estimer les directions. Rappelons tout de même que les sources sont relativement bien séparées, et à égales distances de notre cible. Le résultat peut être vu à la Figure 6.

Ces résultats dépendent bien évidement du nombre de capteurs, de la fréquence des signaux, et du RSBvi.

Dans ce notebook, faisant figure d’introduction au *beamforming,* nous avons bon nombre d’approximations pouvant nous arranger. En réalité, les sources ne seront pas toujours bien séparées spatialement, mais surtout, notre cible sera en mouvement ! Il a donc fallu définir une fonction qui génèrera une suite de positions et d’angles par rapport aux sources, décrivant un mouvement cohérent. Mon choix s’est porté sur un vecteur vitesse constant vers les x positifs.

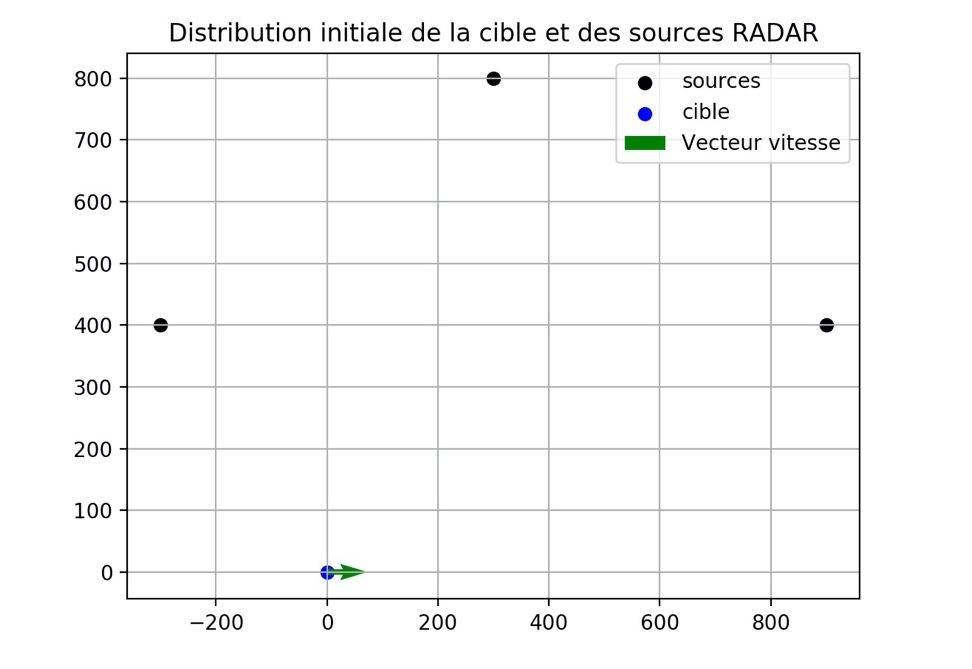


Figure - Affichage des positions des sources et de la cible (axes en m)

Cette situation nous permet de faire varier les angles des sources, mais aussi à deux sources d’être confondues, pour avoir un premier aperçu des performances de nos algorithmes dans un cas non trivial. La détection se passe bien si les sources ne sont pas dans des directions très proches : Au début, les directions sont bien définies. Bien sûr, lorsque les directions de deux (ou plus) RADAR sont les mêmes, il y a ambiguïté, et l’écho trompeur à envoyer pourrait se révéler inefficace, voire incriminant. Cela se voit tout particulièrement avec la méthode la plus simple : *CBF.* Rappelons-nous que nous ne pouvons pas détecter plus de sources que nous avons de capteurs dans notre ULA.

## DoAmouvement

Après l’introduction du mouvement dans le précédent, un nouveau notebook a été créé pour pouvoir se concentrer sur les nouveaux problèmes pouvant être rencontrés avant d’envoyer notre écho trompeur vers nos sources. Quel est le comportement de notre détection lorsque deux cibles sont confondues, et comment quantifier d’éventuels phénomènes ?

Pour agrémenter ce notebook, j’ai choisi de générer les signaux par des *ULA* de dix émetteurs, formant des *classic beamformer.* Si nous nous en tenons à la Figure 3, nous pouvons exprimer le déphasage d’un émetteur à l’autre comme étant : , et le signal émis sera :

Où

* **s** est le vecteur d’état de chaque émetteur à l’instant t
* correspond au signal utile (carré de haute fréquence)
* Le cosinus est le signal « porteur »
* est un bruit additif

Ici, la situation initiale change légèrement : nous allons faire bouger les sources RADAR. En pratique, cela se traduit par deux avions équipés AWACS[[20]](#footnote-12) effectuant une manœuvre de séparation à l’approche de notre cible. L’idée est de récupérer les données de cette manœuvre, d’appliquer nos algorithmes de *DoA*, et de tirer des conclusions.

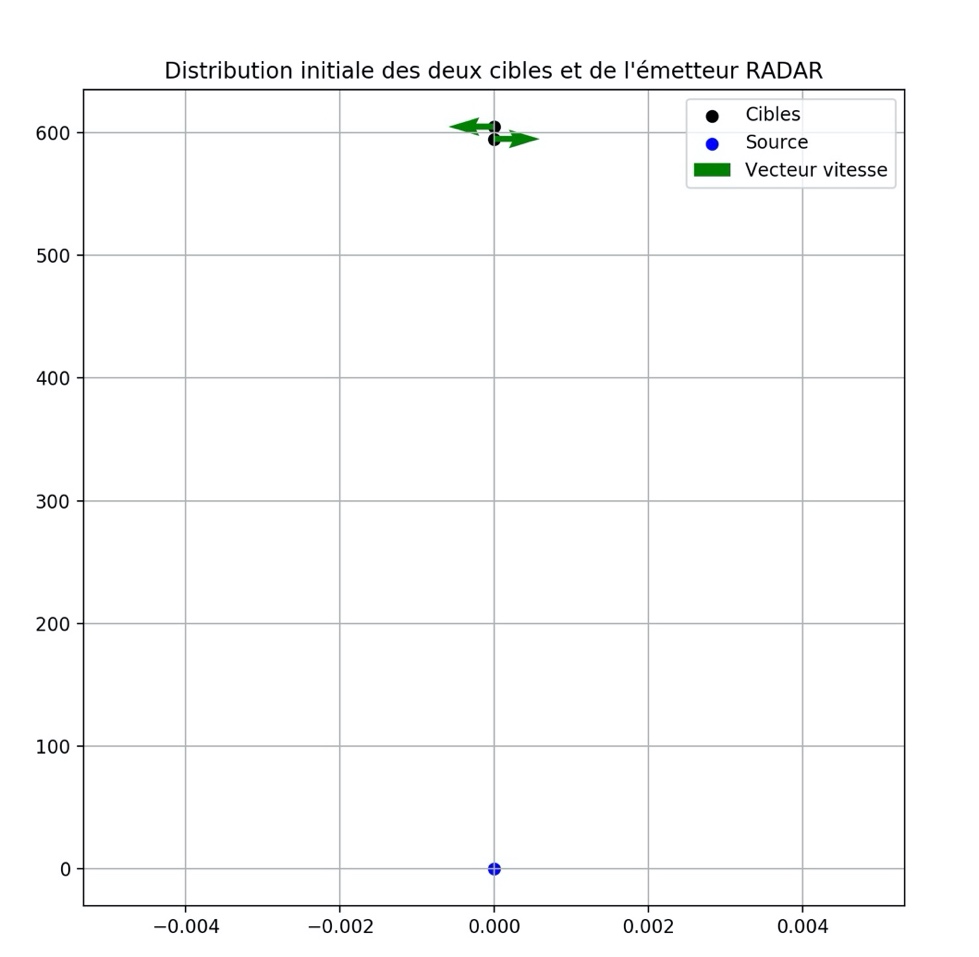


Figure – Affichage des positions des sources et de la cible (axes en m)

Après avoir récupéré et traité les signaux, j’ai affiché les résultats du beamforming classique. C’est avec cette méthodes que les lobes sont les plus distincts, et possède une caractéristique permettant de lever l’incertitude quant à la présence d’une deuxième source dans la même direction : la largeur à mi-hauteur du lobe. Lorsque les sources sont bien séparées angulairement, les lobes associés à leur détection et à l’estimation de leur position ont une largeur semblable. La distance n’influe que peu sur la largeur du lobe, mais plutôt sur l’amplitude maximale.

En lançant l’animation de l’évolution de la direction d’arrivée des signaux par CBF avec les paramètres préétablis, nous distinguons bien la séparation du lobe principal de la fonction 𝑃𝐶𝐵𝐹 en deux lobes. Avant de se séparer en plusieurs pics, on a une augmentation de la largeur du lobe. On peut mesurer cette augmentation en calculant la largeur à mi-hauteur du pic, ce qui est fait par la fonction HMFW[[21]](#footnote-13).

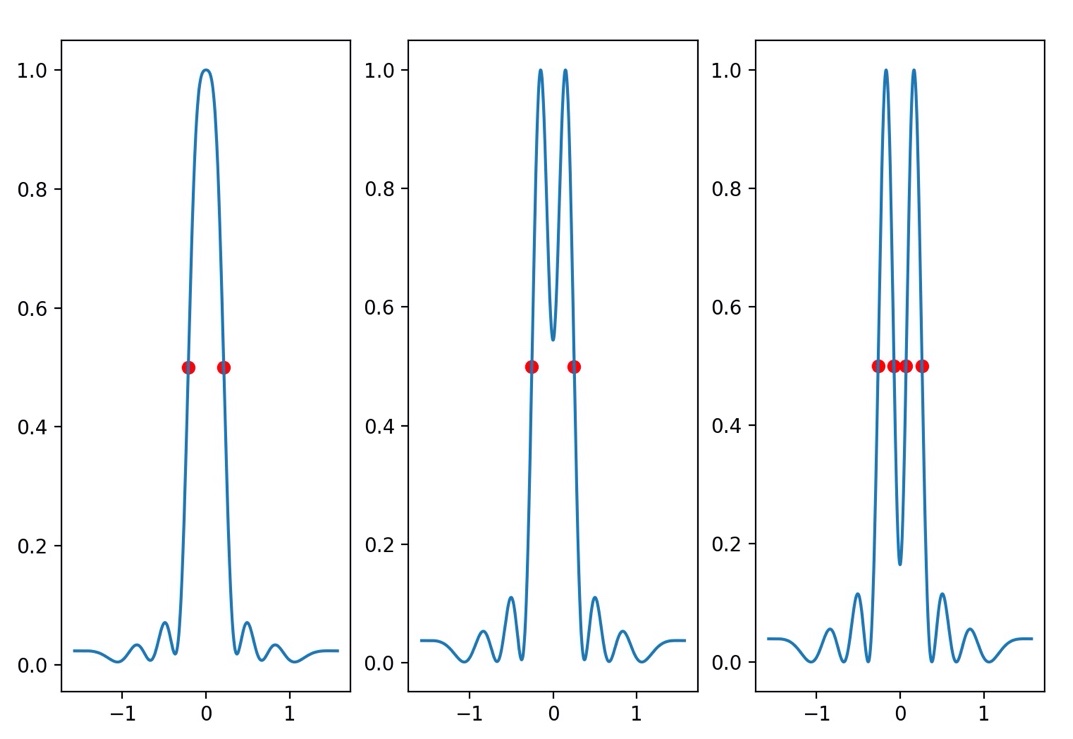


Figure - Séparation du lobe principale et influence sur la largeur à mi-hauteur

J’ai ensuite fait en sorte que les cibles ne soient pas confondues spatialement, afin de vérifier que la distance n’avait pas d’effet sur la largeur des lobes. Il s’avère que les lobes sont bien de la même taille lorsque les sources sont finalement séparées angulairement, mais la méthode CBF va d’abord détecter un changement de pic brusque dans la direction de la cible la plus proche avant que les lobes ne soient totalement séparés.

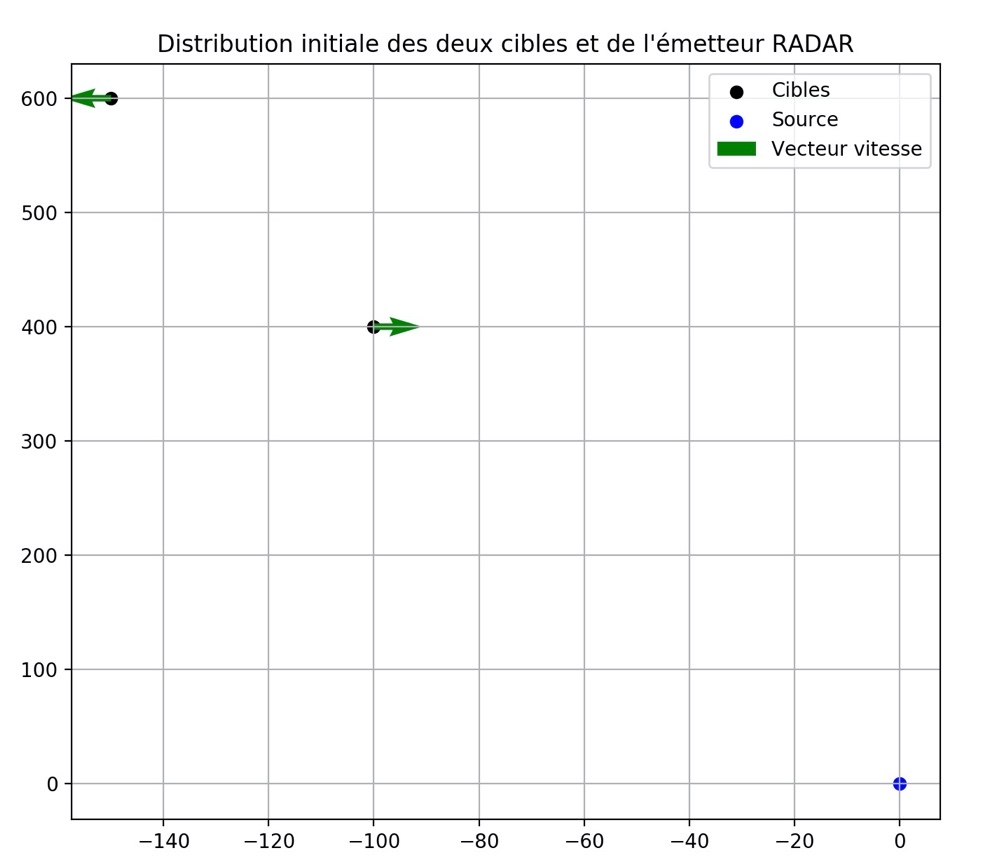


Figure - Affichage des positions des sources et de la cible (axes en m)

Avec le CBF, nous pouvons conclure que la largeur à mi-hauteur des lobes constatés, ainsi qu’un changement brusque de position du pic peuvent témoigner d’une ambiguïté quant à la présence d’autres sources. En fonction de la situation, et de comment le risque et le coût sont quantifiés, nous pouvons introduire une règle de décision précisant la marche à suivre.

Concernant les autres méthodes, les pics sont bien plus précis et beaucoup plus fins : il est donc inutile d’attendre quoi que ce soit de MVDR ou de MUSIC.

Après s’être assuré que les directions pouvaient être estimées convenablement, il ne reste plus qu’à émettre l’écho dans ces directions.

## Simulation

Le 13 Août 2020, le Pr. Neyt m’a envoyé pour la première fois une description précise de ce qu’il attendait de mes simulations. Il m’a donc fait parvenir un code Matlab®[[22]](#endnote-9) simulant des antennes électroniques en présence d’interférences décorrélées. Un des points clés du stage était de voir ce qu’il se passerait si les interférences sont corrélées.

Nous aurons un système mobile, et deux systèmes RADAR. Les deux systèmes RADAR envoient chacun un signal ( et respectivement), ces signaux étant corrélés. Au niveau de notre système mobile, nous recevons . Par rapport à chaque RADAR, système mobile est respectivement à .

Le but ici est d'étudier la matrice , qui est calculée à partir de ce qui est reçu.

La structure de cette matrice dépend de plusieurs facteurs :

* Il y a-t-il du bruit ?
* Les signaux composants s0 sont-ils corrélés ?

Si la structure est simple, comme c'est le cas si les signaux s1 et s2 ne sont pas corrélés, alors les directions d'arrivées peuvent être trouvées aisément. En complexifiant la structure, nous rendons l'obtention des *DoA* plus compliquée.

La matrice R, lorsque les sources ne sont pas corrélées, s’exprime comme ceci : .

Cependant, R se complique si les sources sont corrélées.

Pour la simulation, j’ai placé deux sources respectivement à 20° et 40°.

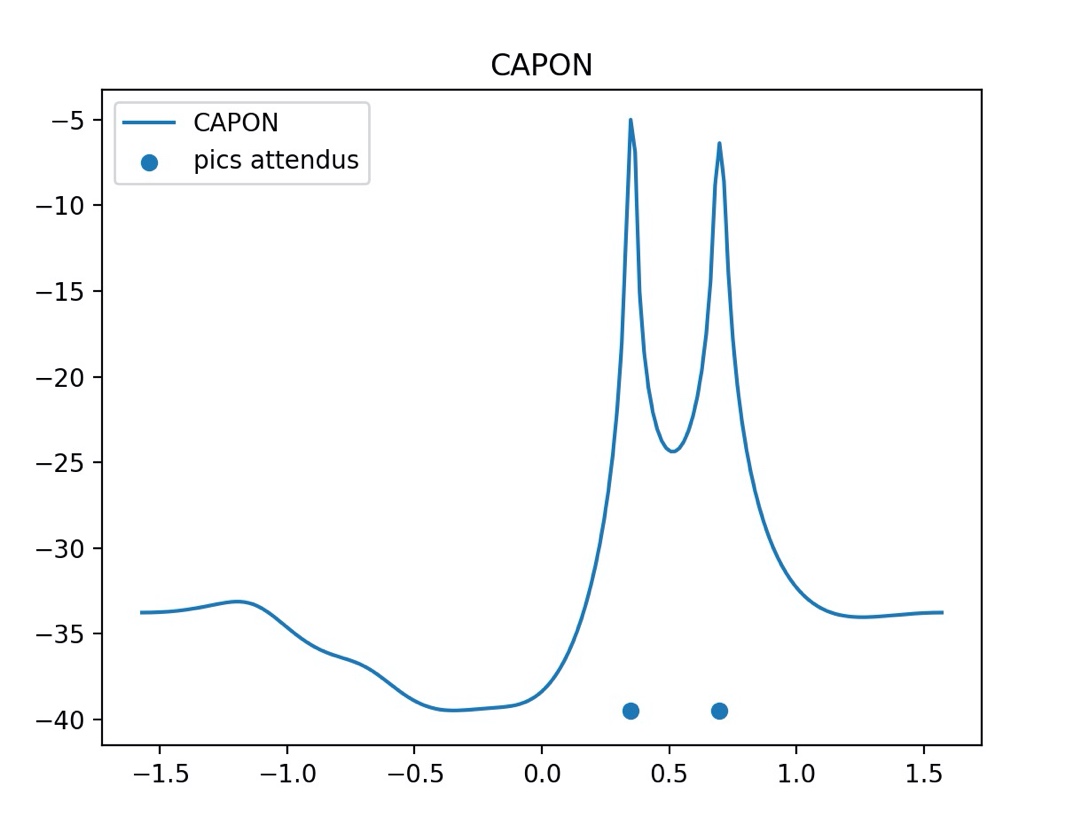


Figure - Estimation des directions d'arrivées de deux signaux non corrélés (angles en rad)

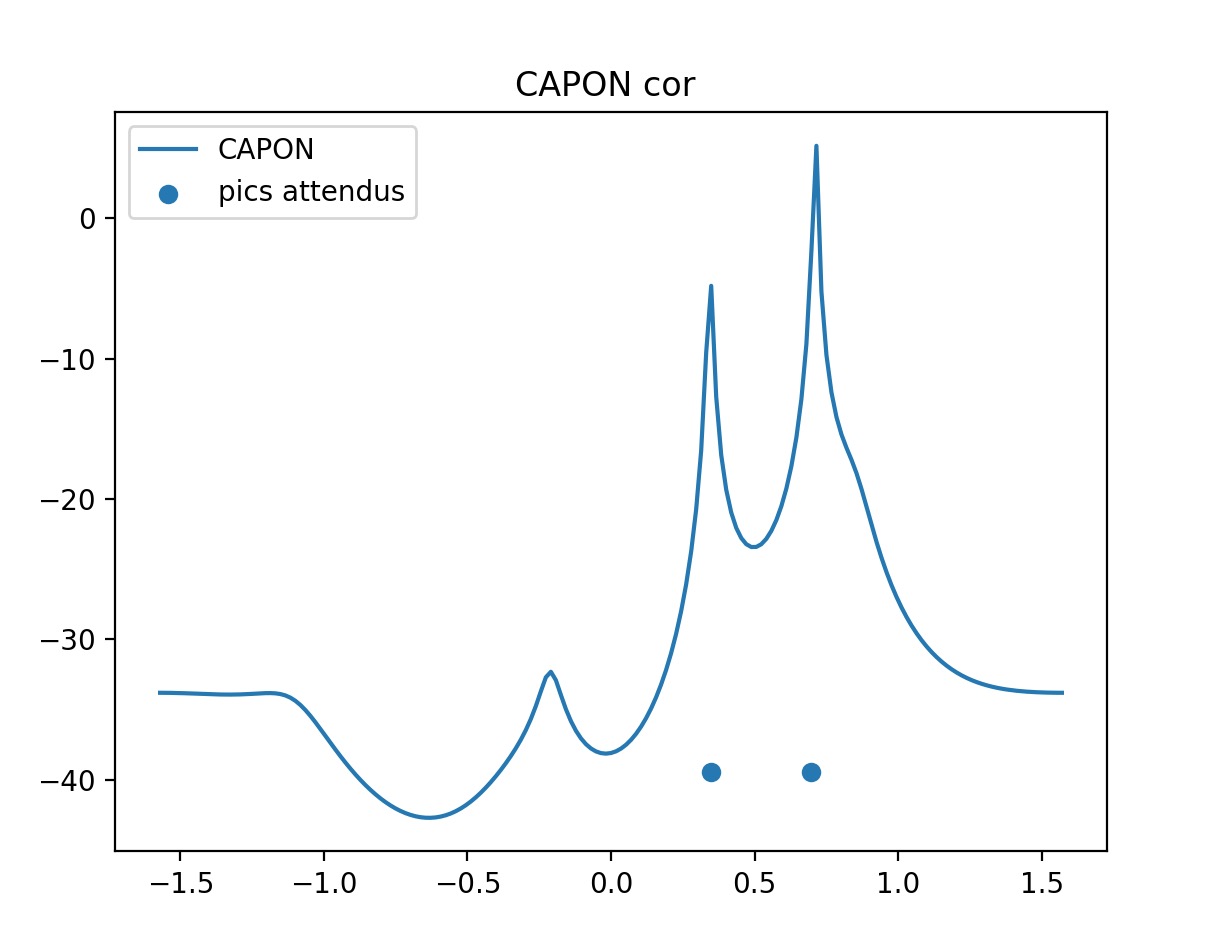


Figure - Estimation des directions d'arrivées de deux signaux corrélés (angles en rad)

Nous nous rendons bien compte qu’en complexifiant la matrice de covariance R, on obtient plus difficilement les directions d’intérêt. En effet, nous voyons dans un premier temps que l’amplitude des pics n’est pas la même, alors que les sources sont à la même distance. De plus, un troisième pic fait son apparition, et vient potentiellement perturber l’estimation des directions d’intérêt en induisant un faux-positif.

# Conclusion

Malgré le contexte sanitaire et les impossibilités de déplacements, je suis reconnaissant d’avoir pu effectuer mon stage en télétravail. La conception d’un banc de contre-mesure n’a pas pu se faire faute de matériel, cependant la découverte de la formation de faisceaux, et la mise en contexte d’un tel système m’ont beaucoup apporté. Cette expérience en autonomie, et la découverte de nouvelles notions m’ont permis de prendre un recul certain sur les notions acquises en deuxième année, de mettre en exergue mes lacunes, mais aussi de me préparer pour ma troisième et dernière année à l’ENSTA Bretagne.

# Table des figures

[Figure 1 : École Royale Militaire de Belgique 7](#_Toc52718110)

[Figure 2 - Logo de Jupyter, outil utilisé durant le stage 9](#_Toc52718111)

[Figure 3 - Opérateur RADAR sur la frégate Chevalier Paul en 2014 - Marine Nationale® 12](#_Toc52718112)

[Figure 4 - Différence de chemin entre deux capteurs successifs - 'Signaux et Sytèmes', Prof. Dr Ir Xavier Neyt 13](#_Toc52718113)

[Figure 5 - Illustration de la directivité des ULA 14](#_Toc52718114)

[Figure 6 - Équivalence des conditions Transformée de Fourier / DoA : Sources localization with sensor array, C. Vanwynsberghe - A. Drémeau 15](#_Toc52718115)

[Figure 7 - Illustration des différentes méthodes de DoA pour 3 sources 17](#_Toc52718116)

[Figure 8 - Diagrammes de rayonnement d'un ULA à 5 capteurs dans des directions données 19](#_Toc52718117)

[Figure 9 - Comparaison des erreurs relatives sur les directions des sources, entre les trois méthodes décrites 20](#_Toc52718118)

[Figure 10 - Affichage des positions des sources et de la cible (axes en m) 20](#_Toc52718119)

[Figure 11 – Affichage des positions des sources et de la cible (axes en m) 22](#_Toc52718120)

[Figure 12 - Séparation du lobe principale et influence sur la largeur à mi-hauteur 23](#_Toc52718121)

[Figure 13 - Affichage des positions des sources et de la cible (axes en m) 23](#_Toc52718122)

[Figure 14 - Estimation des directions d'arrivées de deux signaux non corrélés (angles en rad) 25](#_Toc52718123)

[Figure 15 - Estimation des directions d'arrivées de deux signaux corrélés (angles en rad) 25](#_Toc52718124)

# Annexes

## Annexe 1 – Suivi du travail effectué

*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.

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

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).

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, et les faire fonctionner avec le reste de la simulation.

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

*Semaine 4 : (du Lundi 22 Juin au Dimanche 28 Juin)*

Dans un second notebook noté DoAmouvement.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). »

Ce que j’observe, sur des vitesses de déplacement de la cible assez faibles, c’est un grossissement de la taille du pic de détection. On peut quantifier cette taille et sa variation en donnant la largeur à mi-hauteur du lobe.

Il faut que je commence à m’intéresser à l’amplitude du signal d’entrée. Peut-être que la largeur du lobe n’y est pas corrélée, mais plutôt directement liée à la présence de plusieurs sources. Ce qui est sûr, c’est que la méthode MUSIC ne se prête pas à ça. Voyons tout de même les effets observés.

*Semaines 5 et 6 : (du Lundi 29 Juin au Dimanche 12 Juillet)*

J’ai essayé de simplifier mon code, de le rendre moins complexe en temps d’exécution, mais aussi plus dense : j’ai utilisé un fichier python BeamformingMethods.py pour agglomérer toutes mes fonctions crées jusque-là.

J’ai eu beaucoup de bugs dû à ma piètre connaissance de Jupyter : ces semaines m’ont permis de me familiariser avec quelques astuces, mais aussi à mieux mettre en page les markdowns, avec de vraies équations, plus claires.

*Semaines 7 et 8 : (du Lundi 13 Juillet au Dimanche 26 Juillet)*

Après un retour du Pr. Neyt, j’essaie de commenter un maximum les notebooks créés, car peu compréhensibles selon lui. Il est fait mention du fait que je m’éloigne des expressions mathématiques : outre le fait qu’il est difficile de ne pas traiter de manière discrète quoi que ce soit sur python, je n’ai pas forcément utilisé les mêmes équations, ni la même formulation que le Pr., ce qui explique surement ce quiproquo.

Tout de même, je m’interrogeais sur mon choix initial d’utiliser python.

*Semaine 9 : (du Lundi 27 Juillet au Dimanche 2 Août)*

J’ai commencé à explorer d’autres pistes : dans les logiciels très utilisés dont je peux obtenir une license, il y a Matlab avec Simulink. Évidemment, je ne peux pas utiliser la plateforme NI, celle-ci étant payante, et je n’ai pas leurs boitiers d’acquisition/génération de signaux.

Le site de MathWork étant bien fourni, j’ai pu trouver des exemples de beamforming, essentiellement sonore. Malheureusement simulink, sans exemple concrets, était trop difficile à cerner.

*Semaine 10 : (du Lundi 3 Août au Dimanche 10 Août)*

Pause pour coïncider avec les congés de ma conjointe.

*Semaines 11 et 12 : (du Lundi 11 Août au Vendredi 21 Août)*

Dernier retour en date du Pr. Neyt, à partir duquel je créé le notebook Simulation, dans lequel j’essaie de prendre au maximum ses notations, à partir du fichier matlab qu’il m’a fourni.

Finalisation des notebooks, et commit sur le repository Github fait.

## Annexe 2 – Notebooks et fichier python

[GitHub – Stage Assistant Ingénieur](https://github.com/MaxENSTAB/Stage2A/)

## Annexe 3 – Ressources Matlab pour le notebook Simulation

%%%% antenne.m

N=5;

l=0.3;

d = l/2;

theta = 0/180\*pi;

dphi = 2\*pi\*d\*sin(theta)/l;

n = (0:N-1)';

s = exp(j\*dphi\*n);

y = s;

theta1 = 20/180\*pi;

dphi1 = 2\*pi\*d\*sin(theta1)/l;

s1 = exp(j\*dphi1\*n);

theta2 = 40/180\*pi;

dphi2 = 2\*pi\*d\*sin(theta2)/l;

s2 = exp(j\*dphi2\*n);

%R = s1\*s1' + s2\*s2' + 0.001\*eye(N);

R = zeros(N);

K=1000;

for k=1:K

a1 = randn(1) + j \* randn(1);

a2 = randn(1) + j \* randn(1);

nn = a1 \*s1 + a2 \* s2 + 0.001\*(randn(N,1)+j\*randn(N,1)) + a1\*s2 + a2\*s1;

R = R + nn \* nn';

end

R = R/K;

y = s + s1 + s2;

theta = [-90:90]/180\*pi;

for k=1:length(theta)

dphi = 2\*pi\*d\*sin(theta(k) )/l;

s = exp(j\*dphi\*n);

w = inv(R)\*s;

T(k) = abs(w'\*y)^2;

SNR(k) = s'\*inv(R)\*s;

end

polar(theta,T)

figure

plot(theta/pi\*180, 10\*log10( SNR ));

%%%% adaptant.m

function adaptant

l = 0.3; %Lambda

d = l/2;

N = 5; % Number of sensors (must be uneven)

dphi = 2\*pi\*d/l;

% Electronic antenna, steering weight given by s0

s0 = getsteeringvector(N, dphi, 0);

plotdiagram(N, dphi, s0)

% Adaptive antenna, steering weight given by s0, interference in s1

s1 = getsteeringvector(N, dphi, 30/180\*pi);

R = s1\*s1' + 0.01\*eye(N);

w = inv(R)\*s0;

plotdiagram(N, dphi, w)

% Adaptive antenna, steering weight given by s0, interference in s1 and s2, uncorrelated

s2 = getsteeringvector(N, dphi, 50/180\*pi);

L = 1000;

phi1 = randn(L,1)+j\*randn(L,1);

phi2 = randn(L,1)+j\*randn(L,1);

sn = s1\*phi1' + s2\*phi2';

R = sn\*sn'+0.01\*eye(N);

w = inv(R)\*s0;

plotdiagram(N, dphi, w)

plotsnr(N, dphi, R)

%+++++++++++++++++++++++++++++++++++++++++++++++++++

function plotdiagram(N, dphi, s0 )

theta = (-90:90)'/180\*pi;

A = zeros(length(theta),1);

for k = 1:length(theta)

s = getsteeringvector(N, dphi, theta(k));

A(k) = s0'\*s;

end

plot( theta/pi\*180, abs(A));

figure

polar( theta, abs(A));

%+++++++++++++++++++++++++++++++++++++++++++++++++++

function plotsnr(N , dphi, R )

theta = (-90:90)'/180\*pi;

A=zeros(length(theta),1);

for k=1:length(theta)

s=getsteeringvector( N, dphi, theta(k));

A(k) = s'\*inv(R)\*s;

end

plot( theta/pi\*180, 20\*log10(abs(A)));

figure

polar( theta, abs(A));

%+++++++++++++++++++++++++++++++++++++++++++++++++++

function s = getsteeringvector(N, dphi, theta)

n = (1:N)';

s = exp(j\*n\*dphi\*sin(theta));

# Bibliographie

|  |  |
| --- | --- |
| [1] | X. Neyt, «Signaux et Sytèmes,» 2018, p. 228 à 230. |
| [2] | A. D. C. Vanwynsberghe, «Source Localization with sensor array,» 2020. |

# Glossaire technique

1. Jupyter Notebook : Calepin contenant à la fois du code exécutable et du texte de type *Markdown.* [↑](#endnote-ref-1)
2. RADAR Cross Section : Aussi appelée Surface Équivalente RADAR, elle représente l’importance relative de la surface de réflexion d'un faisceau électromagnétique qu’un objet provoque. [↑](#endnote-ref-2)
3. ULA : Un Uniform Linear Array est une succession de capteurs disposés linéairement et à distances égales. Dans notre cas, les capteurs utilisés sont des antennes RADAR. [↑](#endnote-ref-3)
4. OCA : Les Ingénieurs de Études et Techniques de l’Armement et les Ingénieurs de l’Armement constituent les Officiers des Corps de l’Armement. [↑](#endnote-ref-4)
5. Prof. Dr. Ir, correspond aux trois titres suivants dans le système Belge : Professeur, Docteur, et Ingénieur. [↑](#footnote-ref-1)
6. Élie Nguyen-Huy-Thuy et François Schmidt ont effectué leur stage au même département CISS, ainsi qu’avec le même tuteur. [↑](#footnote-ref-2)
7. Github : Entreprise américaine développant l’outil de suivi de version et de stockage éponyme, le dossier où sont enregistrés les fichiers utilisés s’appelant un repository. [↑](#endnote-ref-5)
8. [Repository Github](https://www.github.com/MaxENSTAB/Stage2A) [↑](#footnote-ref-3)
9. Ici, « éclairer » est une expression militaire correspondant à « viser avec une pulsation électromagnétique ». [↑](#footnote-ref-4)
10. RSB : Rapport signal sur bruit. [↑](#endnote-ref-6)
11. Python est un langage Open Source, complet et facile à appréhender grâce à ses nombreuses bibliothèques. [↑](#footnote-ref-5)
12. DoA : *Directions of Arrival,* littéralement directions d’arrivée d’un signal. [↑](#endnote-ref-7)
13. [HackRF Git](https://github.com/mossmann/hackrf/wiki/HackRF-One) [↑](#footnote-ref-6)
14. *Radio Frequencies*: Fréquences Radio [↑](#footnote-ref-7)
15. LabView® [↑](#footnote-ref-8)
16. Un snapshot = un vecteur de mesures [↑](#footnote-ref-9)
17. Steering Vector : Vecteur représentant l'ensemble des déphasages subis par une onde plane, évalués au niveau d'*ULA.* [↑](#endnote-ref-8)
18. Soit le nombre de sources éclairant la cible [↑](#footnote-ref-10)
19. [↑](#footnote-ref-11)
20. *Airborne Warning And Control System* : Système de détection et de commandement aéroporté [↑](#footnote-ref-12)
21. Half Maximum Full Width : largeur à mi-hauteur du pic [↑](#footnote-ref-13)
22. Matlab : MATLAB est un langage de script utilisé à des fins de calculs scientifiques. Il rend très pratique la manipulation de matrices et de vecteurs [↑](#endnote-ref-9)