Systèmes de télécommunications aéronautiques - BE

Institut Polytechnique des Sciences Avancées (IPSA)

Simulation d'un radar sol-sol

Réalisé par :

Florian Alaux, Guilhem Perret-Bardou

Encadrants :

M. Ortega

Année académique : 2024-2025

09 mai 2025

Table des matières

1 Introduction

Ce projet nous permettra de découvrir et de comprendre le fonctionnement d'un radar à travers une approche progressive et interactive. Il est divisée en deux parties interconnectées : un radar unidimensionnel et un radar bidimensionnel.

Dans un premier temps, nous travaillerons sur un radar unidimensionnel pour analyser la détection des cibles en fonction de leur distance. Nous simulerons le signal d'écho et étudierons les phénomènes d'ambiguïté liés à la résolution et aux limitations du système, comme la portée maximale sans ambiguïté. Nous explorerons également l'influence de paramètres clés, tels que la période et la fréquence de répétition des impulsions (PRT et PRF), et mettrons en oeuvre des algorithmes de détection.

Dans la seconde partie, nous passerons à un radar bidimensionnel avec balayage angulaire. Cette étape nous permettra d'analyser la détection des cibles en azimut et de mieux comprendre la formation des images radar. Nous serons amenés à évaluer les performances du système face au bruit et aux fausses alarmes, en appliquant des techniques d'estimation adaptées.

Ce projet nous donnera une vision complète du traitement du signal radar, en alliant simulation numérique et expérimentation interactive. A travers cette expérience, nous d´evelopperons nos compétences en analyse de signaux et en modélisation radar, tout en explorant les défis techniques liés à ces systèmes.

2 Partie 1 : Implémentation du radar unidimensionnel

2.1 Implémentation du pulse rectangulaire (*generate_pulse.m*)

Pour commencer ce projet, nous avons implémenté la génération d'un pulse rectangulaire dans la fonction *generate_pulse*. Cela consistait à définir le vecteur temporel t en calculant la durée totale du signal ($t\_total = PRT * num\_pulses$) et en générant les points temporels avec un pas de 1/Fs. Ensuite, nous avons identifié les indices de début ($start\_idx$) et de fin (end_idx) de chaque impulsion en utilisant les paramètres Fs, Tp, et la période de répétition PRT, pour positionner correctement les impulsions dans le temps. Enfin, nous avons généré le pulse rectangulaire en assignant la valeur 1 aux indices correspondants dans le vecteur tx_signal pour le cas '$rectangulaire$'.

2.2 Implémentation de l'écho (*simulate_echo.m*)

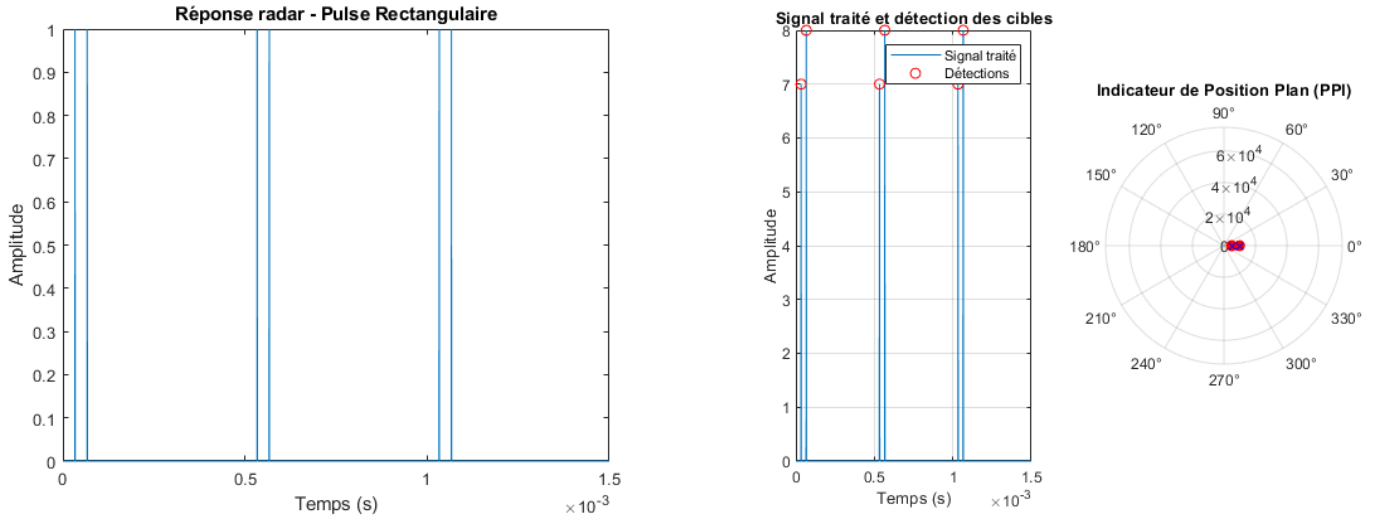
Le but de cette fonction est d'implémenter les échos générés par chaque cible et de génerer le signal qui parvient au récepteur. Pour ce faire : On a commencé par ajouter la periode de répetition des impulsions $\frac{1}{PRF}$ et la fréquence. Ensuite on a calculé la durée des pulsations et leur amplitude moyenne. Puis le temps de retard τ et grâce à ce τ on peut convertir ce temps de retard en indice d'échantillonage en faisant $2 \times F_s$. Pour continuer, on a découpé notre largeur de cible en un échantillonnage. Ensuite, comme dans la partie précédente on a défini notre *startidx* et notre *endidx*. Et enfin pour terminer cette section on a généré l'écho grâce à la fonction *pulsamplitude* multiplié par une matrice *ones*.

2.3 Implémentation du traitement du signal à la réception (*process_signal.m*)

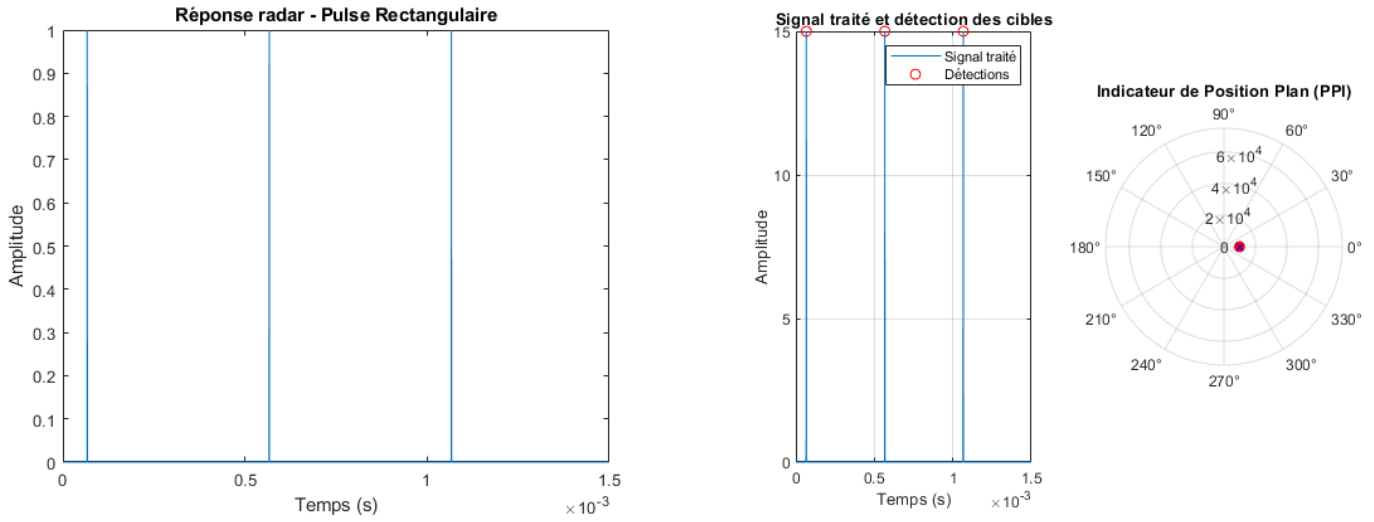
Dans cette fonction, nous allons appliquer les connaissances acquises lors du cours théorique afin de localiser les impulsions réfléchies. Pour ce faire, nous allons utiliser, à la réception, le filtre adapté. Le filtre adapté est un outil crucial en radar, conçu pour maximiser le rapport signal sur bruit (SNR) lors de la réception des impulsions réfléchies. En inversant dans le temps et en conjuguant le signal transmis, puis en le corrélant avec le signal reçu, il permet une compression du pulse, améliorant ainsi la résolution temporelle. Ce mécanisme optimise la détection des cibles, même dans des conditions de faible SNR, facilitant leur localisation précise.
Pour génerer le filtre adapté, on a utilisé les fontions $fliplr.m$ et *conv.m* La fonction $fliplr.m$ en MATLAB inverse l'ordre des colonnes d'une matrice. La fonction *conv.m* réalise la convolution entre deux vecteurs, utilisée pour appliquer le filtre au signal.
Pour la détection des impulsions, dans une seconde instance, il faut mettre en œuvre un algorithme de détection. Pour cela, il faut deux éléments : d'une part, le signal en sortie du filtre adapté, et d'autre part, un seuil qui indique si la puissance reçue correspond à une impulsion ou non. Pour réaliser la détection, nous allons utiliser la fonction $findpeaks.m$, à laquelle nous passerons en arguments la valeur $MinPeakHeight$ et la valeur du seuil.
L'utilisation du filtre adapté génère un décalage associé au traitement du signal. Ce décalage est appelé retard de groupe et, dans le contexte d'un filtre FIR, il est égal à la moitié de la durée du filtre. Nous avons appliqué cette correction en utilisant *time_corrected*.
Enfin, la distance des cibles doit être comparée par rapport à la distance maximale d'ambiguïté. Nous avons définit une distance d'ambiguïté maximale, ensuite nous avons utilisés *detected_targets* qui est le modulo entre la distance calculée et la distance d'ambiguïté maximale définie précédement. Ici, le modulo sert à ramener les distances dans une plage définie qui correspond à celle où les cibles sont détectées par le RADAR.

2.4 Evaluation du code (pulse rectangulaire)

Lorsqu'on lance le code on obtient les résultats suivants :



Tout d'abord, nous pouvons regarder notre réponse radar et constater que nous avons bien un pulse rectangulaire, avec une période correspondant à T_p. En zoomant sur le graphique du signal traité, nous pouvons remarquer que les pics sont très étroits, indiquant une résolution temporelle fine. Celle-ci est cohérente avec une impulsion rectangulaire courte. Les changements d'amplitudes que nous pouvons repérer dans le signal traité montre bien que nous avons plusieurs cibles détectées. Nous voyons effectivement ici que les deux cibles présentes à 5000m et 10000m sont bien détectées comme nous le voyons avec les deux cercles rouges.



Cette fois, lorsque nous plaçons les cibles à 10 000m et 10 020m on remarque qu'il y a qu'une seule detection de cible. En effet les cibles sont tellement proches qu'elles sont confondues dans le signal. Cela peut être en partie due au fait que les cibles se superposent à cause de leur largeur aussi importanteque la distance qui les sépare.

Ceci ne représente que la partie *Pulse rectangulaire*. Désormais nous allons étudier le pulse modulé, qui est censé nous permettre d'améliorer la précision, même pour deux cibles très proches.

2.5 Implémentation du pulse modulé (*generate_pulse.m*)

Un pulse modulé est un signal utilisé dans les systèmes radar pour améliorer la résolution et la capacité à détecter des cibles rapprochées. Contrairement à un pulse rectangulaire simple, qui a une largeur de bande fixe, un pulse modulé peut être configuré de manière à avoir des caractéristiques qui lui permettent de mieux discriminer les cibles, même celles qui sont très proches les unes des autres. L'une des formes les plus courantes de modulation est la modulation en fréquence (ou chirp), où la fréquence du signal varie linéairement au cours du temps. Cela permet d'étendre la largeur de bande effective du signal, ce qui augmente la résolution du radar.
Un pulse modulé de type chirp peut être exprimé par l'équation suivante :

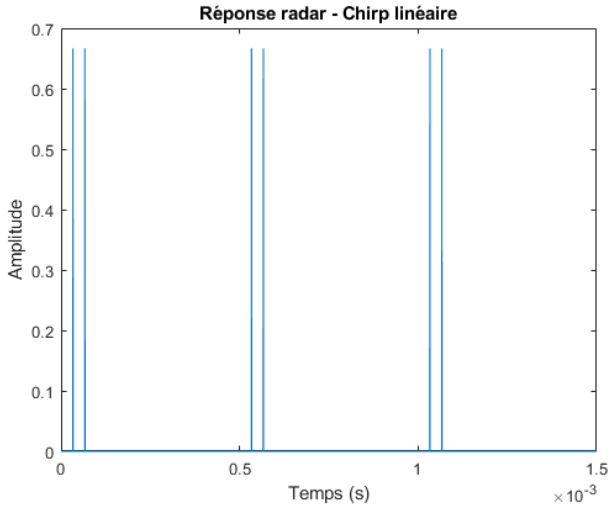
$$P(t) = Acos(2\pi(f_0 t + B/(2T_p)t^2)) \tag{1}$$

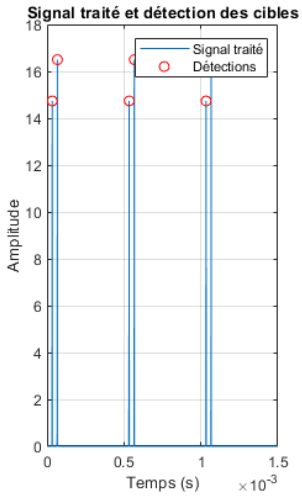
Où :
- A est l'amplitude du signal
- f_0 est la fréquence initiale du pulse,
- B est la bande passante du signal
- T_p est la durée du pulse.

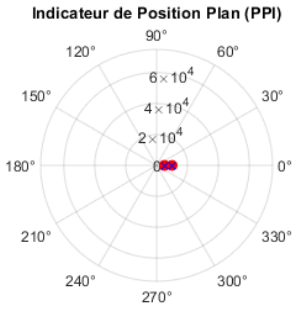
La modulation de fréquence permet de compresser l'impulsion dans le domaine temporel tout en augmentant la bande passante du signal, ce qui améliore la capacité du radar à distinguer des cibles proches.
Cela signifie que même si les cibles sont situées à des distances similaires, un pulse modulé permet de mieux les localiser grâce à la meilleure résolution temporelle obtenue par l'élargissement de la bande passante. En résumé, l'utilisation d'un pulse modulé, et plus spécifiquement d'un chirp, est essentielle pour les radars modernes, car elle permet d'améliorer la précision de la détection des cibles, particulièrement lorsqu'elles sont proches les unes des autres.
Pour générer le pulse modulé, on a complété notre code avec l'équation (1) avec les données suivantes :
- $B = 10MHz$
- $f_0 = 0Hz$
- τ est l'axe du temps.

2.6 Evaluation du code (pulse modulé)

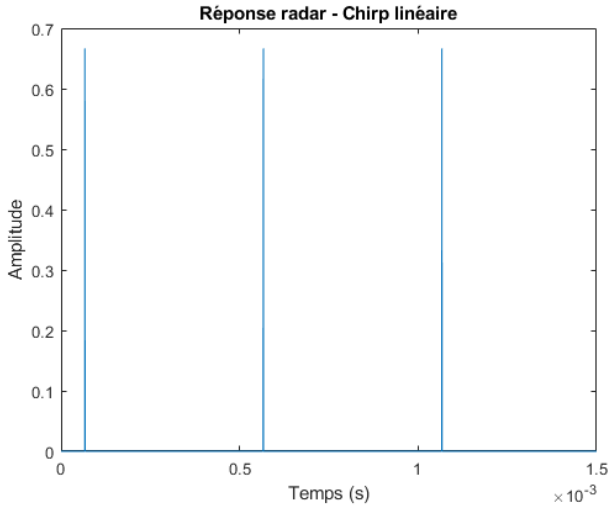
En lançant le code on a pu vérifier qu'il n'y avait pas d'erreurs lors de la compilation ce qui nous a permis d'obtenir les résultats suivants :

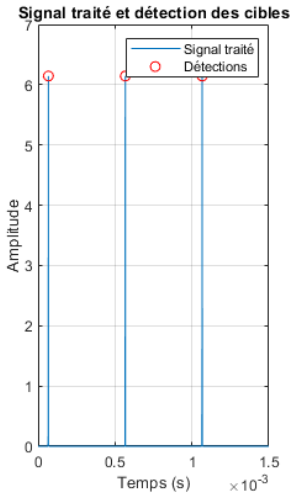


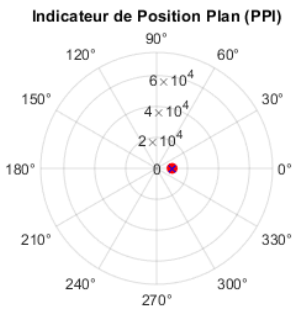




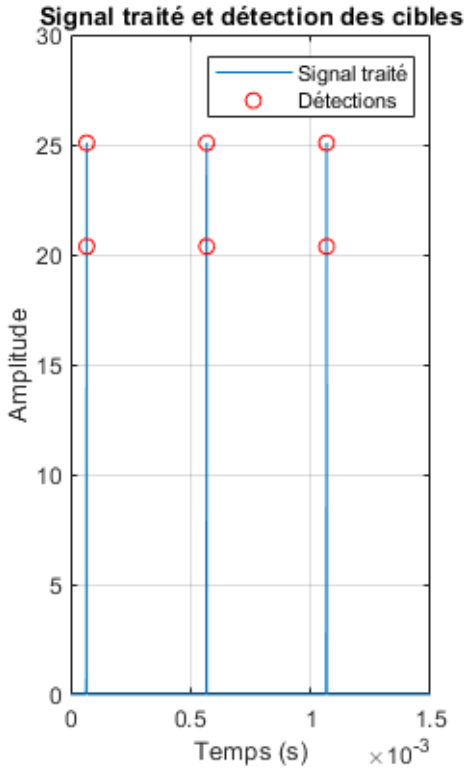
Avec deux cibles à 5 000m et 10 000m, on remarque que la version avec le pulse modulé détecte correctement les cibles, avec les même remarques que pour pulse rectangulaire et même plus précisement que celui-ci.

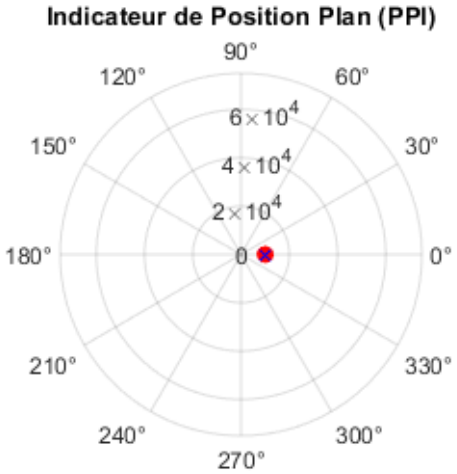






Cette fois-ci en testant avec les cibles avec 10 000m et 10 020m on remarque qu'une seule cible est repérée. Comme pour le signal rectangulaire, c'est très probablement dû à la largeur des cibles qui est égale à 20m, les deux cibles sont donc confondues. C'est pour vérifier cela que nous avons adapté la distance des cibles et refait un test.

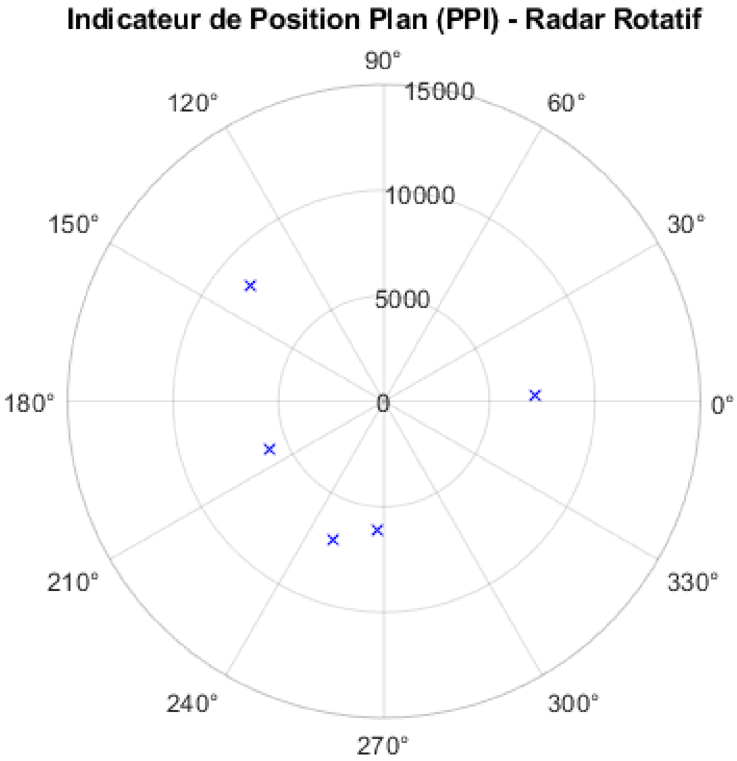




Cette fois ci, avec les cibles à 10 000m et 10 050m, on peut effectivement repérer les deux cibles précisément.
Sur cette partie, nous avons pu découvrir les pulses rectangulaire et modulé. Nous avons surtout pu voir l'importance des seuils. En particulier, le seuil sur le pulse rectangulaire a dû être largement diviser pour détecter les cibles. Au contraire, avec le pulse modulé nous avons dû le remonter et jouer légèrement avec l'amplitude de la fonction chirp.
Mais jusque là, nos signaux ont toujours été simples à détecter. Nous allons maintenant voir un nouveau type de radar pour améliorer les détections et la précision.

3 Partie 2 : Implémentation du radar bidimensionnel avec et sans bruit.

3.1 Evaluation du code

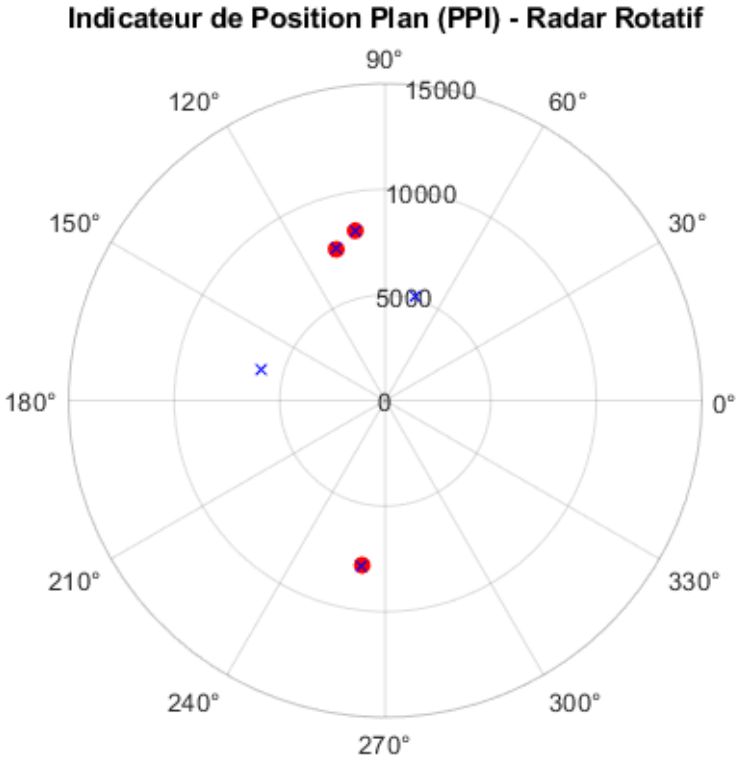


On remarque qu'aucune cible n'est détectée. C'est parce que dans ces conditions, le seuil de détection est trop haut, MATLAB renvoie des erreurs disant qu'aucune valeur de dépasse $MinPeakHeight$:

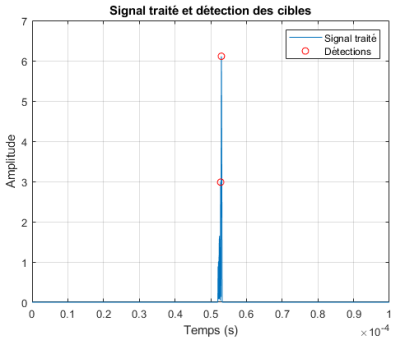
```
Warning: Invalid MinPeakHeight. There are no data points greater than MinPeakHeight.
> In findpeaks>removePeaksBelowMinPeakHeight (line 535)
In findpeaks (line 162)
In process_signal_rx_2D (line 19)
In function_radar_2D (line 49)
Warning: Invalid MinPeakHeight. There are no data points greater than MinPeakHeight.
> In findpeaks>removePeaksBelowMinPeakHeight (line 535)
In findpeaks (line 162)
In process_signal_rx_2D (line 19)
In function_radar_2D (line 49)
Warning: Invalid MinPeakHeight. There are no data points greater than MinPeakHeight.
> In findpeaks>removePeaksBelowMinPeakHeight (line 535)
In findpeaks (line 162)
In process_signal_rx_2D (line 19)
In function_radar_2D (line 49)
Warning: Invalid MinPeakHeight. There are no data points greater than MinPeakHeight.
> In findpeaks>removePeaksBelowMinPeakHeight (line 535)
In findpeaks (line 162)
In process_signal_rx_2D (line 19)
In function_radar_2D (line 49)
Warning: Invalid MinPeakHeight. There are no data points greater than MinPeakHeight.
> In findpeaks>removePeaksBelowMinPeakHeight (line 535)
In findpeaks (line 162)
In process_signal_rx_2D (line 19)
In function_radar_2D (line 49)
```

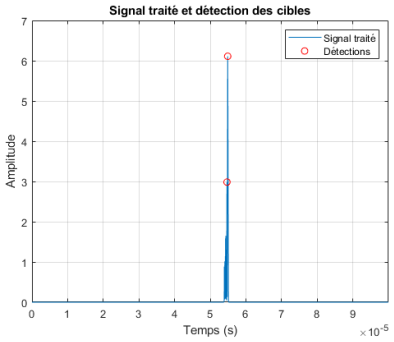
On voit ici cinq fois la même erreur, pour cinq cibles non détectées. En effet, elles peuvent être trop loin et hors de la zone de détection, ou alors notre seuil est trop élevé et empêche qu'elles soient détectées. Cela se confirme quand nous changeant sa valeur pour vérifier cela :

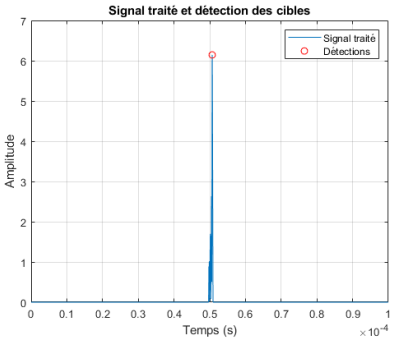
```
threshold = E_s/10;
```



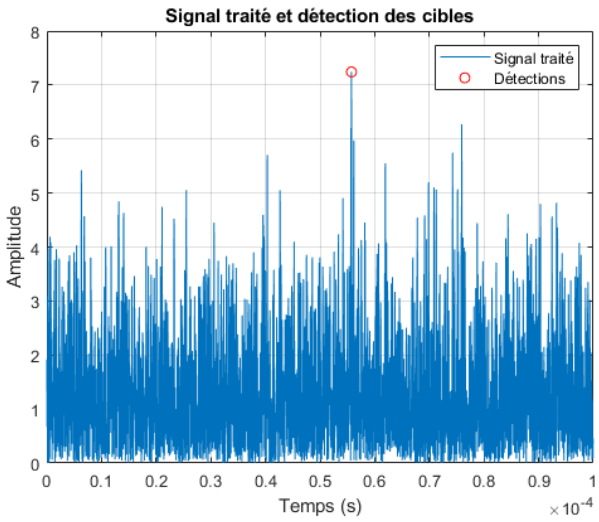
Cette fois-ci trois cibles détectées sur cinq, et nous pouvons voir leurs signaux traités :

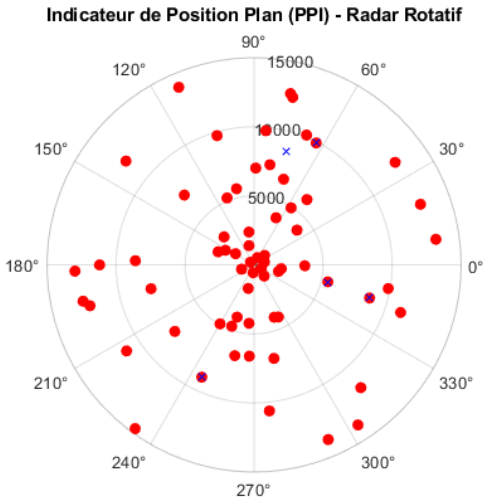






Mais diminuer le seuil global n'est pas une bonne solution. En effet, en étant trop bas, le bruit pourrait finir par être détecté comme une cible. Dans notre cas, il n'y avait pas de bruit donc diminuer le seuil n'était pas génant. Mais tout est très différent si nous mettons simplement $P\_bruit = 0.1$.





Malgré notre seuil retourné à $E\_s/4$, nous pouvons voir que de très nombreuses fausses détections sont apparues seulement à cause du bruit, de plus, les vraies cibles ne sont pas forcément détectées.
C'est pour cela que nous allons maintenant parler des radar CFAR.

3.2 Implémentation d'un radar CFAR.

Un radar CFAR (Constant False Alarm Rate) est une technique utilisée pour détecter des cibles dans un environnement bruité tout en maintenant un taux d'alarme fausse constant. Cela signifie que le radar ajuste dynamiquement son seuil de détection en fonction du bruit ambiant dans chaque région de détection, afin de réduire les faux positifs (détections erronées). Le radar CFAR évalue le bruit local autour de la cible suspecte et adapte le seuil de détection pour qu'il soit suffisamment bas pour détecter la cible, mais suffisamment élevé pour éviter de détecter le bruit comme une cible. Cela permet d'améliorer la détection dans des environnements complexes et à faible rapport signal sur bruit (SNR).
Dans le cours théorique, nous avons effectué la dérivation pour être capables de calculer ce seuil de manière théorique sous certaines hypothèses, en particulier que la puissance du signal est normalisée et que le signal modulé est un signal complexe. Dans notre implémentation, aucune de ces hypothèses n'est valable, car la modulation utilisée est un signal réel et nous n'avons pas normalisé le signal.
Sous ces hypothèses, en l'absence de cible, le signal en sortie du filtre adapté est un bruit gaussien réel $n(t) \sim \mathcal{N}(0, P_{\text{bruit}})$. Le filtre effectue une corrélation avec l'impulsion $s(t)$ d'énergie $E_s = \sum s^2(t)$. La sortie $Y = \sum n(t)s(t)$ est gaussienne, avec :

$$\mathbb{E}[Y] = 0, \quad \text{Var}(Y) = \sum s^2(t) \cdot P_{\text{bruit}} = E_s \cdot P_{\text{bruit}}.$$

Ainsi, $Y \sim \mathcal{N}(0, \sigma^2)$, où $\sigma^2 = E_s \cdot P_{\text{bruit}}$.
L'énergie en sortie, $Z = Y^2$, suit une loi exponentielle (chi-deux à 1 degré de liberté) de densité :

$$f_Z(z) = \frac{1}{2\sigma^2} e^{-z/(2\sigma^2)}, \quad z \geq 0.$$

La probabilité de fausse alarme est :

$$P_{FA} = \mathbb{P}(Z > T) = \int_T^\infty \frac{1}{2\sigma^2} e^{-z/(2\sigma^2)}\, dz = e^{-T/(2\sigma^2)}.$$

Le seuil T s'exprime alors comme :

$$T = -2\sigma^2 \ln(P_{FA}).$$

En termes de chi-deux, $Z/\sigma^2 \sim \chi^2(1)$, et $\mathbb{P}(W > \tau) = P_{FA}$ donne $\tau = \chi^2_{\text{inv}}(1 - P_{FA}, 1)$. Ainsi :

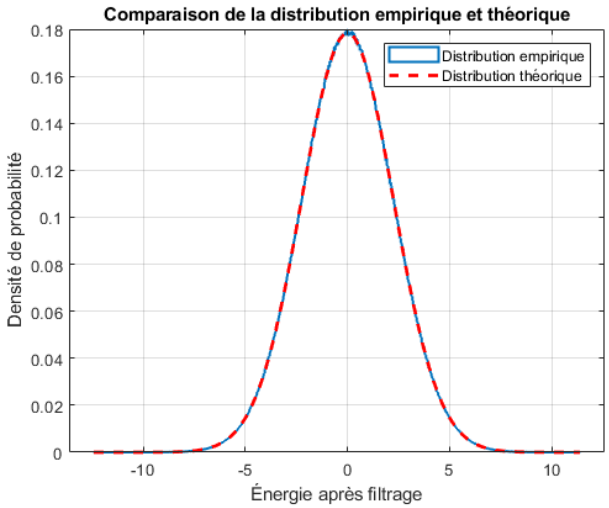
$$T = \sigma^2 \cdot \chi^2_{\text{inv}}(1 - P_{FA}, 1) = E_s \cdot P_{\text{bruit}} \cdot \chi^2_{\text{inv}}(1 - P_{FA}, 1).$$

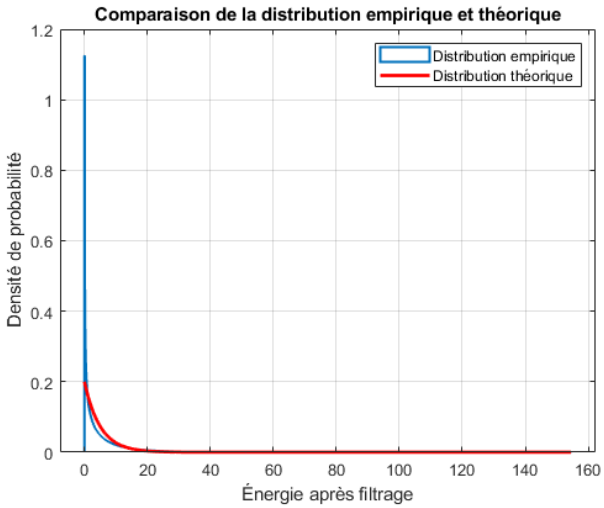
Ce seuil est utilisé dans notre implémentation pour ajuster dynamiquement le seuil de détection dans le radar CFAR.

3.3 Implémentation de la simulation Monte Carlo

A partir de ce nouveau calcul de seuil, nous afin de valider l'implémentation du détecteur CFAR dans un environnement bruité, nous avons réalisé une simulation Monte Carlo avec 10^7 essais pour estimer empiriquement la probabilité de fausse

alarme (PFA). Cette simulation se fait à partir de la génération de bruit gaussien, auquel nous appliquons un filtre adapté basé sur la forme du pulse transmis.





La première figure montre que la sortie du filtre adapté (produit scalaire entre bruit et pulse) suit bien une loi normale centrée, ce qui est cohérent avec un bruit gaussien réel traité par un filtre linéaire. La seconde figure montre la distribution de l'énergie en sortie du filtre, c'est à dire le carré du signal, qui suit une loi exponentielle. Dans les deux cas, nous pouvons remarquer que la distribution empirique est conforme au modèle théorique. Tout cela semble confirmer que notre modèle est valide, donc que le seuil optimal est vérifié. Nous sommes conforté dans cette idée avec les sortie dans la console MATLAB :

```
Probabilité de fausse alarme théorique: 0.00010
Probabilité de fausse alarme empirique (Monte Carlo): 0.00010
Différence relative: 0.00200
```

4 Conclusion

En conclusion, ce projet nous a permis de découvrir et comprendre et utiliser un système radar avec des pulse rectangulaire et modulé. Nous avons pu apprendre à gérérer des impulsions, traiter les signaux reçus pour détecter des cibles à distance mais aussi analyser les résultats grâce aux graphiques et diagrammes PPI. Nous avons pu mettre en opposition les pules rectangulaire et modulé mais aussi voir les avantages du chirp. Enfin, nous avons pu valider notre calcul du seuil de détection par une simulation de Monte-Carlo.