THÉORIE DES NOMBRES ET RADARS

FRANÇOIS JAULIN

Résumé

Dans ce qui suit, on présente le radar Doppler, ainsi que le problème qu'il rencontre à ses début en 1943. La résolution de ce problème revient à déterminer une permutation dite « parfaite » qui vérifie une certaine propriété. On résout ce problème en se plaçant sur des corps finis, grâce à la donnée d'un élément primitif qui engendre le corps ou en utilisant le logarithme de Zech.

1. INTRODUCTION

HISTORIQUE

Au début de la seconde guerre mondiale, les dispositifs radars, utilisés pour repérer des diverses cibles (bateaux, avions, etc), ne sont pas encore très sophistiqués. Pour passer au travers de ces derniers, les cibles lâchent derrière elles des petites pièces métalliques qui brouillent et saturent complètement la réception radar : les radars, qui à l'époque envoient des ondes de fréquence f fixe, sont victime de l'effet Doppler, explicité dans la suite, qui disperse la fréquence initiale des ondes sur plusieurs fréquences et rend impossible la localisation de la cible.

En 1943, les ingénieurs mettent au point un radar qui envoie des ondes de fréquences différentes, et qui permet de faire face à cette éventualité. L'idée consiste ensuite à mesurer dans l'écho, c'est-à-dire les ondes émises par le radar qui nous reviennent après réflexion contre divers obstacles dont la cible, les différences et les modifications de fréquences dues à l'effet doppler qui traduirait alors une vitesse effective de la cible, ou bien due à un retard qui traduirait alors la distance de la cible au radar.

Cependant très tôt dans la pratique, les opérateurs se retrouvent confrontés à l'ambiguïté suivante : ces radars dits radars Doppler confondent par exemple une cible stationnaire à 10 km avec une cible à 9 km se déplaçant avec une grande vitesse d'approche. Cette confusion peut évidement s'avérer fatale.

Dans la suite, nous allons essayer de mieux comprendre mathématiquement la confusion puis essayer d'apporter une solution à ce problème, tout en gardant à l'esprit que ce qui nous intéresse, c'est la distance et la vitesse de la cible par rapport au radar.

Date: Mardi 19 Juillet 2005.

1.1. PRÉLIMINAIRES.

1.1.1. *Définitions.*

Définition 1 (groupe symétrique)**.** *Soit $n \in \mathbb{N}^*$. On note. $\mathcal{S}_n$ le groupe des permutations de $\{1,\dots,n\}$ (muni de la loi de composition). Le groupe $\mathcal{S}_n$ est appelé* groupe symétrique *d'indice n. Si $s \in \mathcal{S}_n$ on note $s = \left(\begin{smallmatrix} 1 & 2 & \cdots & n \\ s(1) & s(2) & \dots & s(n) \end{smallmatrix}\right)$.*

Définition 2 (matrice de permutation)**.** *Soit $\sigma \in \mathcal{S}_n$ une permutation. On associe à cette permutation une matrice $M(\sigma) = (m_{ij})_{(i,j)\in[1;n]^2}$ où $m_{ij} = \begin{cases} \times & si\ \sigma(j)=i \\ - & si\ \sigma(j)\neq i \end{cases}$.*

$M(\sigma)$ s'appelle la matrice de permutation *σ ou s'il n'y a pas de risques de confusion la matrice de permutation.*

Par exemple, à la permutation $\sigma=\left(\begin{smallmatrix} 1&2&3&4&5&6 \\ 3&2&6&5&1&4 \end{smallmatrix}\right) \in \mathcal{S}_6$ correspond la matrice de permutation:

$$M(\sigma) = \begin{pmatrix} - & - & - & - & \times & - \\ - & \times & - & - & - & - \\ \times & - & - & - & - & - \\ - & - & - & - & - & \times \\ - & - & - & \times & - & - \\ - & - & \times & - & - & - \end{pmatrix}$$

1.1.2. *Fonctionnement du radar Doppler.*

On donne ici des caractéristiques du radar Doppler, mais on ne s'intéresse pas dans la suite à sa constitution physique ni à son fonctionnement interne.

Caractéristiques

Le radar Doppler [1] envoie des ondes avec une *période* T (seconde) de manière à ce que les données sur la cible (vitesse et distance) puissent être mise à jour régulièrement.

Il possède aussi une fonction de précision $\Delta r = 2.c.\Delta t$ où c est la célérité de la lumière et Δt correspond à un temps de précision fixé (Δt est souvent de l'ordre d'une fraction de milliseconde dans la pratique).

Sans imposer d'importantes contraintes sur notre dispositif, on définit une relation entre T et Δt : $\mathbf{T} = (\mathbf{p}^{\alpha}-\mathbf{1})\mathbf{\Delta t}$ où p est un nombre premier et α un entier.

En ce qui concerne la *fréquence*, on suppose pour des raisons pratiques qu'à chaque intervalle de temps, le n$^{\text{ième}}$ commençant au temps t_n défini par:

$$t_n = n.\Delta t, \qquad n = 1, 2, \dots, p^{\alpha} - 2$$

une seule onde de fréquence donnée est envoyée. Toutes les Δt secondes, la fonction qui a une onde envoyée par le radar lui associe sa fréquence prend une nouvelle valeur, puis reste constante pendant Δt.

Le schéma suivant donne un exemple d'évolution possible de la fréquence en fonction du temps.

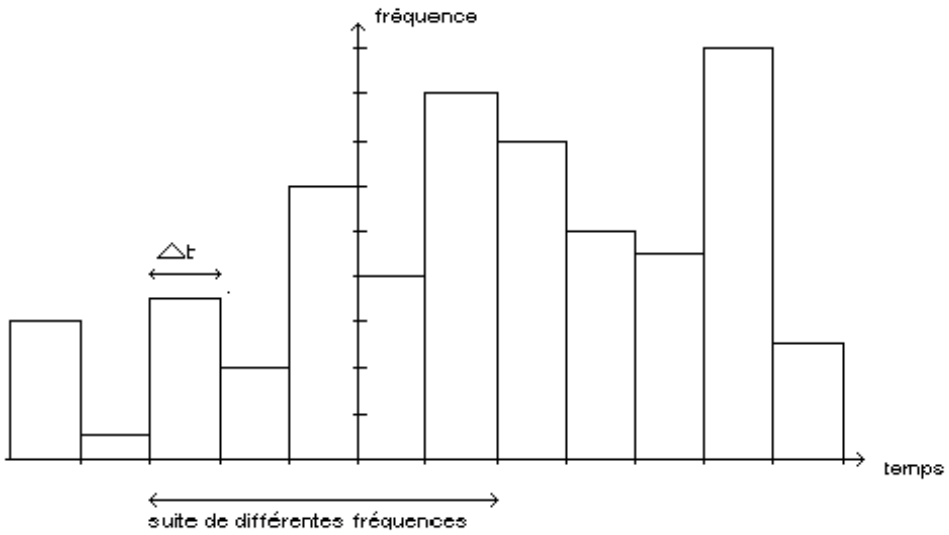


Figure 1 : Exemple d'évolution de la fréquence en fonction du temps

On pourra écrire ce phénomène discret de la façon suivante :

$$f(n) = f(0) + \sigma(n).\Delta f, \text{ où}$$

- $f(0)$ représente la fréquence de la première onde envoyée à t_0 pendant le premier intervalle de temps,
- Δf qui correspond à une fréquence, fixée, qui s'ajoutera un facteur entier $\sigma(n)$ près, à la fréquence initiale $f(0)$,
- σ une permutation de $\mathcal{S}_{p^\alpha-2}$ qui correspond lorqu'elle est prise en n au facteur de modification de la fréquence au n$^{\text{ième}}$ temps et traduit finalement l'évolution de la fréquence.

Ces trois paramètres sont fixés par l'opérateur.

Maintenant que nous avons donné les caractéristiques du radar Doppler, traduisons mathématiquement le problème auquel se retrouvent confrontés les ingénieurs en 1943.

1.2. **PROBLÈME.**

Dans cette partie, on traduit mathématiquement le problème que rencontre le radar Doppler en 1943 mais on ne s'intéresse pas à l'étude des ondes envoyées ni aux modifications qu'elles subissent.

1.2.1. *Hypothèses et « formalités ».*

Effectuons une simulation. On a vu que lorsque l'opérateur souhaite utiliser le radar, il doit, entre autre, disposer d'une permutation.
Fixons une permutation $\sigma \in \mathcal{S}_{p^\alpha-2}$. On suppose qu'on dispose aussi d'une fréquence initiale et d'une fréquence Δf qui s'ajoutera comme on l'a vu précédemment à un facteur près à la fréquence initiale.

Lorsque l'opérateur envoie pendant T secondes un signal constitué d'ondes avec des fréquences associées comme cela a été expliqué dans les préliminaires, il reçoit en retour après un instant donné des ondes qu'il avait envoyé et qui se sont réfléchies sur divers obstacles dont la cible.

Les ondes reçues qui se sont réfléchies sur la cible en mouvement subissent un changement de fréquence dû à l'effet Doppler dont on ne fera pas l'étude mais dont on admettra ce qui suit :

Proposition 1 (Effet Doppler)**.** *Lorsqu'on envoie un signal sinusoïdal de fréquence fixe f_0 vers une cible animée d'un mouvement et dont la vitesse suivant l'axe d'émission est notée v, le signal se réfléchit et subit un changement de fréquence $\Delta f = 2.v.\frac{f_0}{c}$ où c est la célérité de la lumière. [2]*

Pour plus détails, on pourra se référer à la page internet :

http://francoisjaulin.free.fr/Tipe.htm

On fait pour l'étude qui suit deux hypothèses :

- On suppose que le temps Δt est suffisamment petit pour que la vitesse de la cible soit considérée comme constante entre t et $t + (p^\alpha - 2).\Delta t = t + T$. ($\Delta t$ ou T sont des paramètres que l'utilisateur peut régler)

- On suppose que le dispositif radar a la possibilité, lorsque qu'il envoie des signaux dans toutes les directions autour de lui, de filtrer les signaux de retour provenant d'une direction particulière, et dans notre cas, la direction de la cible.

On ne s'occupe donc que des ondes qui proviennent de la cible.

Par ailleurs, le réglage fait sur Δt ou T nous permet de faire l'approximation suivante :
comme pendant le temps T (petit), la vitesse de la cible peut être considérée comme constante, l'effet Doppler sur une onde de fréquence donnée f

revient à la multiplier par un facteur constant - en réalité qui dépend de la vitesse mais qui est elle supposée constante - ce qui revient par passage au logarithme à ajouter une constante au logarithme de cette fréquence. Cette constante additive sera significative de la vitesse de la cible. En pratique, il suffira donc de connaître cette constante pour connaître la vitesse de la cible.

On a schématiquement :

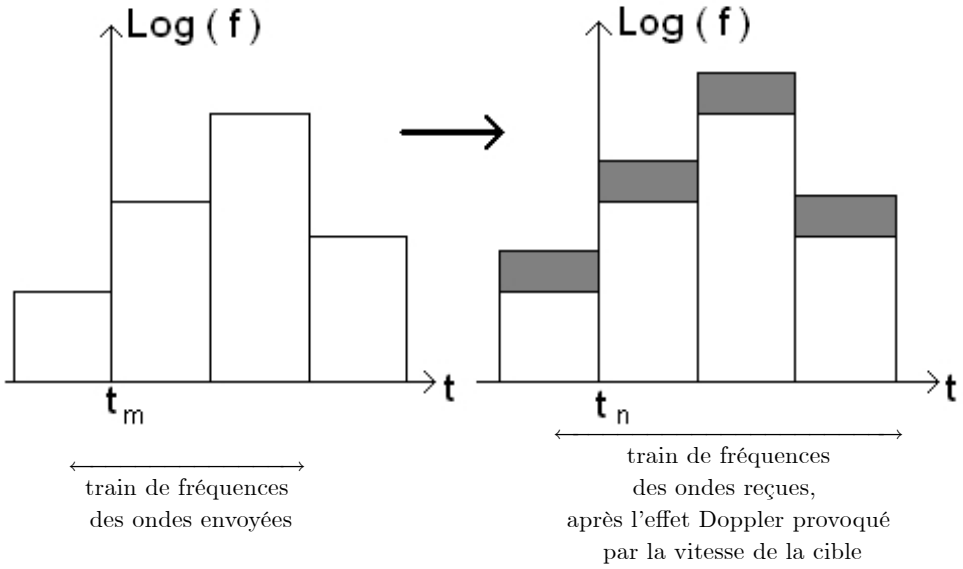


Figure 2 : Effet Doppler

$$f_{retour} = f_{envoie}.(2.\frac{v}{c}) \Rightarrow \log(f_{retour}) = \log(f_{envoie}) \pm \log(2.\frac{v}{c})$$

Le correspond en réalité au signe du projeté de la vitesse de la cible sur l'axe orienté Cible-Radar. + correspond à une cible qui s'approche et − à un cible qui s'éloigne du radar. v est toujours positif dans l'expression avec le log.

C'est ici que la notion de matrice vient prendre tout son sens : comme on l'a vu précédemment, aux signaux d'envoi et de retour peut s'associer une matrice de permutation. On parlera de matrice d'envoi pour parler de la matrice associée au signal d'envoie et plus directement pour désigner la permutation choisie par l'opérateur avant l'utilisation du radar. À partir de là, on associera au signal de retour, dans son intégralité - même quand rien n'est perçu par le radar - une matrice de retour. Cela signifie donc que la matrice de retour commencera temporellement au même moment que la matrice d'envoie, et ses premières colonnes seront remplies de traits (−) jusqu'à la perception par le radar d'une première onde provenant de la cible qui sera alors modélisée par la présence d'une croix (×).

On peut alors constater deux choses :

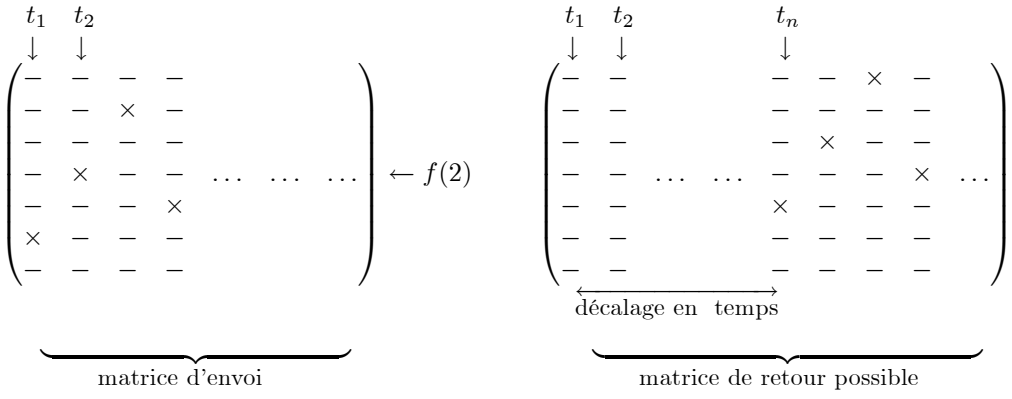
- Tout décalage entre ces deux matrices le long d'une colonne donnée traduit une modification en fréquence de l'onde.
- Tout décalage entre ces deux matrices le long d'une ligne donnée traduit une modification dans le temps.

Or on rappelle ici que c'est justement le décalage en temps qui nous donnera la distance de la cible au radar et le décalage en fréquence (effet Doppler) qui nous donnera la vitesse d'approche de la cible .

$$\left(\frac{\begin{array}{c}\text{décalage}\\ \text{en temps}\end{array}}{\begin{array}{c}\text{vitesse des ondes}\\ \text{émises par le radar}\end{array}} = 2 \times \text{(distance Radar-Cible)} \right)$$

$$\left(\text{décalage en fréquence} = \log(2.\tfrac{v}{c}) \Rightarrow v = \tfrac{c}{2}.e^{\text{décalage en fréquence}}\right)$$

Exemple sans valeurs précises avec le train de fréquence de la Figure 2 :



1.2.2. *Problème dans l'analyse des signaux.*

On aura peut-être déjà entrevu le problème à partir de l'exemple précédent. Reprenons le pour illustrer le problème plus précisément. On peut remarquer dans l'exemple précédent que la matrice de retour possède un motif

$$\begin{pmatrix} - & - & \times & - \\ - & - & - & - \\ - & \times & - & - \\ - & - & - & \times \\ \times & - & - & - \end{pmatrix}$$

commun avec la matrice de permutation. Mais supposons que pour des raisons de pertes diverses, la cible n'ait renvoyé vers le radar que le motif :

$$\begin{pmatrix} - & \times \\ - & - \\ \times & - \end{pmatrix}$$

on est alors incapable, ce motif étant deux fois présent dans la matrice d'envoi, de savoir auquel il correspond, c'est-à-dire de savoir les décalages en temps et en fréquence correspondants (i.e. la vitesse d'approche et la distance).

En effet, on a dans cet exemple deux interprétation possibles :

- soit la cible est à une distance d faible avec une petite vitesse d'approche
- soit la cible est à une distance D ($D > d$) moyennement faible mais avec une vitesse d'approche assez élevée.

Cela pose une certaine ambiguïté qui peut s'avérer fatale pour les utilisateurs qui rappelons le sont en 1943 en plein guerre.
L'idée d'envoyer des ondes avec différentes fréquences était clairement un progrès mais cela a amené un nouveau problème.

Pour résoudre ce problème, il faut trouver un moyen d'envoyer des ondes avec des fréquences différentes de sorte que dans la matrice de permutation (matrice d'envoi), il n'existe aucun motif contenant au moins deux croix ($\times$) qui soit identique.

Mathématiquement, résoudre ce problème revient à trouver des permutations $\sigma \in \mathcal{S}_{p^\alpha - 2}$ qui vérifient pour tout $k \neq 0$ et $n \neq n' \mod (p^\alpha - 1)$ la relation suivante :

$$\sigma(n+k) - \sigma(n) \neq \sigma(n'+k) - \sigma(n')$$

On dira que les permutations qui vérifient cette relation sont des *permutations parfaites*. Ces permutations ne sont pas simples à trouver, et représentent une faible partie de toute les permutations possibles. Il est par conséquent utile et nécessaire de mettre en place une méthode rigoureuse pour obtenir ces permutations.

2. RÉSOLUTION MATHÉMATIQUE

Quelques résultats à admettre

On va, pour résoudre le problème auquel est soumis le radar Doppler, admettre un certain nombre de résultats issus de la théorie des corps finis et qui ne constituent pas le sujet de notre TIPE. On pourra en revanche trouver plus de détails et d'explications sur Internet à l'adresse suivante : www.jenesaispasencore.com

On précise au lecteur avant de rentrer dans les détails que ce qui nous intéresse ici, c'est uniquement de savoir calculer sur les corps finis qui vont être présentés, afin de pouvoir ensuite faire un programme qui calcule des permutations parfaites, et d'en trouver nous même « à la main ».

2.1. PROCÉDÉ ÉLÉMENTAIRE.

Soit p un nombre premier. Le quotient de l'anneau $\mathbb{Z}$ par l'idéal maximal $p\mathbb{Z}$ que l'on notera $\mathbb{F}_p$, est un corps. Le groupe multiplicatif $\mathbb{F}_p^*$ associé est isomorphe à $\mathbb{Z}/(p-1)\mathbb{Z}$. Il admet donc $\phi(p-1)$ générateurs où ϕ est la fonction indicatrice d'Euler.

On a vu que notre radar ne peut être efficace que si l'on dispose d'une permutation $\sigma \in \mathcal{S}_{p^\alpha-2}$ telle que pour tout $k \neq 0$ et $n \neq n' \mod (p^\alpha - 1)$:

$$\sigma(n+k) - \sigma(n) \neq \sigma(n'+k) - \sigma(n') \mod (p^\alpha - 1)$$

2.1.1. *Permutation donnée par un élément générateur de $\mathbb{F}_p^*$.*

On peut facilement trouver une telle permutation dans le cas $\alpha = 1$. Pour cela, on se place dans le groupe des éléments inversibles de $\mathbb{F}_p$, c'est-à-dire $\mathbb{F}_p^*$ et on prend un élément g qui engendre ce groupe - on appelle ces éléments des *éléments primitifs* - et on considère la permutation $\sigma \in \mathcal{S}_{p-1}$ définie par:

$$\begin{array}{c} \{1,\dots,p-1\} \to \mathbb{F}_p^* \\ n \mapsto g^n \end{array}$$

Théorème 1. *Cette permutation est parfaite.*

▲ En effet, supposons qu'il existe k, n, n' vérifiant les hypothèses tels que:

$$g^{n+k} - g^n = g^{n'+k} - g^{n'}$$

En factorisant, on a :

$$g^n(g^k - 1) = g^{n'}(g^k - 1)$$

Et comme g est primitif dans $\mathbb{F}_p^*$, il est d'ordre $p-1$. Or par hypothèse, $k \neq 0 \mod (p-1)$, donc on a $g^k \neq 1$. On simplifie et on obtient :

$$g^n = g^{n'}$$

Ce qui n'est possible que si $n \equiv n' \mod (p-1)$.

Cela contredit donc les hypothèses. □

2.1.2. *Inconvénient de la méthode.*

Une discussion avec plusieurs mathématiciens m'a montrée que cette méthode n'était pas très pratique, car il est difficile de trouver des éléments g primitifs dans $\mathbb{F}_p^*$. La méthode empirique qui consiste à prendre des nombres au hasard puis les tester pour voir s'ils sont primitifs n'est pas satisfaisante. En effet, le calcul qui suit montre qu'on ne peut jamais être sûr d'avoir la chance de tomber sur un nombre primitif, parce que leur nombre varie fortement.

Raisonnement :

On cherche la proportion d'éléments primitifs de $\mathbb{F}_p^*$. Il y en a $\phi(p-1)$ en tout parmi p. On se demande si lorsque p est grand, cela a une limite ou non. Cela revient à se demander si $n \longmapsto \frac{\phi(n)}{n}$ a une limite.

- Si n est premier, $\phi(n) = n-1$ donc $\frac{\phi(n)}{n} \longrightarrow 1$ lorsque $n \longrightarrow +\infty$.
- Si on note p_i le ième nombre premier, on a :

$$\phi\left(\prod_{i=1}^{k} p_i\right) = \prod_{i=1}^{k} (p_i - 1)$$

donc :

$$\frac{\phi\left(\prod\limits_{i=1}^{k} p_i\right)}{\prod\limits_{i=1}^{k} p_i} = \prod_{i=1}^{k} \frac{(p_i-1)}{p_i} = \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

Or l'inverse de ce produit est :

$$\prod_{i=1}^{k} \frac{1}{\left(1 - \frac{1}{p_i}\right)}$$

Lorsque $k \longrightarrow +\infty$, ce produit tend vers l'infini, car en développant les termes $\frac{1}{\left(1-\frac{1}{p_i}\right)}$,toutes les puissances de p_i sortent, et ensuite on peut comparer ces termes à la somme $\sum \frac{1}{n}$ qui tend vers l'infini. (On retrouve d'ailleurs le théorème du cours : $\sum \frac{1}{p_i}$ diverge).

Donc :

$$\frac{\phi\left(\prod_{i=1}^{k} p_i\right)}{\prod_{i=1}^{k} p_i} \longrightarrow 0 \text{ lorsque } k \longrightarrow +\infty$$

Cela montre que 0 et 1 sont des valeurs d'adhérence de $n \longmapsto \frac{\phi(n)}{n}$, ce qui par conséquent entraîne l'impossibilité de savoir exactement la proportion d'éléments primitifs parmi $(p-1)$ éléments.

2.2. PROCÉDÉ AMÉLIORÉ : LE LOGARITHME DE ZECH.

Il existe d'autres corps finis, ce sont les corps de décomposition des polynômes de $\mathbb{F}_p[X]$. On note $\mathbb{F}_{p^\alpha}$ le corps de décomposition de $X^{p^\alpha} - X$ sur $\mathbb{F}_p$.

La théorie des extensions de corps est complexe, et ce n'est pas le propos de notre TIPE. Nous admettons simplement l'existence de corps à p^α éléments, et nous voulons juste savoir calculer dans ce corps. On admet à cet usage les résultats suivants :

Théorème 2. *Le corps $\mathbb{F}_{p^\alpha}$ est isomorphe au quotient de l'anneau $\mathbb{F}_p[X]$ des polynômes à coefficients dans $\mathbb{F}_p$ par l'idéal maximal engendré par un polynôme Q irréductible de degré α à coefficient dans $\mathbb{F}_p$. Ainsi donc, l'ensemble des polynômes de degré inférieur ou égal à $(\alpha-1)$ à coefficients dans $\mathbb{F}_p$ fournit un système de représentants de $\mathbb{F}_{p^\alpha}$.*

En pratique :

- On identifiera les éléments de $\mathbb{F}_{p^\alpha}$ à leur représentant (de degré inférieur ou égal à $\alpha-1$), en pratique, par des vecteurs à valeur dans $\mathbb{F}_p$.
- La somme se calcule alors facilement, comme somme de deux polynômes.
- L'opposé est tout aussi facile, il suffit de prendre le polynôme ayant des coefficients opposés.
- Pour le produit, on calcule simplement dans l'anneau quotient : on effectue d'abord le produit des deux représentants dans $\mathbb{F}_p[X]$, puis, le représentant associé au résultat (du produit) sera le reste de la division euclidienne du résultat par le polynôme Q - ce qui revient, en fait, à calculer dans $\mathbb{F}_p[X]$ modulo Q.
- L'inverse de l'élément X de $\mathbb{F}_p[X]/(Q)$ se calcule simplement en faisant ce qui suit :

Si $Q = \sum_{i=0}^{\alpha} a_i X^i$, (calculer modulo Q revient à poser $Q = 0$) on a $Q = 0$, soit:

$$-a_0 = X \times \left(\sum_{i=1}^{\alpha} a_i X^{i-1} \right)$$

Ainsi donc, $\left(-\sum_{i=1}^{\alpha} \frac{a_i}{a_0} X^{i-1} \right)$ est l'inverse de X dans $\mathbb{F}_p[X]/(Q)$, soit dans $\mathbb{F}_{p^\alpha}$.

Exemple avec $\mathbb{F}_{2^3}$:

Sans chercher à savoir comment on obtient des polynômes irréductibles, car cela n'est pas le propos de notre TIPE, on vérifie facilement que par exemple le polynôme $X^3 - X - 1$ est irréductible sur $\mathbb{F}_2[X]$ et il est de degré 3.

Le calcul dans $\mathbb{F}_2[X]$ modulo $X^3 - X - 1$ signifie par exemple que :

$$(X^2 - 1) \times (X + 1) = X^3 + X^2 - X - 1 = (X + 1) + X^2 - X - 1 = X^2$$

Enfin, la représentation sous forme de vecteurs des éléments de $\mathbb{F}_{2^3}$ est la suivante :

éléments de $\mathbb{F}_{2^3}$											représentation vectorielle		
X^0	=	0	×	X^2	+	0	×	X	+	1	0	0	1
X^1	=	0	×	X^2	+	1	×	X	+	0	0	1	0
X^2	=	1	×	X^2	+	0	×	X	+	0	1	0	0
X^3	=	0	×	X^2	+	1	×	X	+	1	0	1	1
X^4	=	1	×	X^2	+	1	×	X	+	0	1	1	0
X^5	=	1	×	X^2	+	1	×	X	+	1	1	1	1
X^6	=	1	×	X^2	+	0	×	X	+	1	1	0	1

2.2.1. *Permutation donnée par le Logarithme de Zech.*

Définition 3. *Soit g un élément de $\mathbb{F}_{p^\alpha}$. Pour $n \in \mathbb{N}$, on définit le* logarithme de Zech *de n, noté $\mathcal{Z}_g(n)$ comme l'unique entier qui vérifie :*

$$g^{\mathcal{Z}_g(n)} = 1 - g^n \text{ (dans } \mathbb{F}_{p^\alpha} \text{ bien sûr).}$$

On peut maintenant généraliser ce qui précède dans le cas où $\alpha \neq 1$. Soit un élément g primitif de $\mathbb{F}_{p^\alpha}$. On considère alors la permutation $\sigma \in \mathcal{S}_{p^\alpha - 1}$ définie par :

$$\{1, ..., p^\alpha - 1\} \to \mathbb{F}_{p^\alpha}$$

$$n \longmapsto \mathcal{Z}g(n)$$

Théorème 3. *Cette permutation est parfaite.*

▲ En effet, si la propriété est fausse,

il existe : n, n' et k tels que :

$k \neq 0 \mod (p^{a-1})$, $n \neq n' \mod (p^{a-1})$, et $\sigma(n+k) - \sigma(n) = \sigma(n'+k) - \sigma(n')$.

Soit

$$\frac{g^{(\mathcal{Z}g(n+k))}}{g^{(\mathcal{Z}g(n))}} = \frac{g^{(\mathcal{Z}g(n'+k))}}{g^{(\mathcal{Z}g(n'))}}$$

et par définition du logarithme de Zech :

$$\frac{1-g^{(n+k)}}{1-g^n} = \frac{1-g^{(n'+k)}}{1-g^{n'}}$$

alors

$$(1-g^{(n+k)}) \times (1-g^{n'}) = (1-g^{(n'+k)}) \times (1-g^n)$$

et en développant :

$$g^{(n+k)} + g^{n'} = g^{(n'+k)} + g^n$$

soit

$$g^n(g^{k-1}) = g^{n'}(g^{k-1})$$

et comme $k \neq 0 \mod (p^{a-1})$, on a : $g^{k-1} \neq 0$ donc $g^n = g^{n'}$

soit $n = n' \mod (p^{a-1})$.

Cela contredit donc les hypothèses. □

Exemple avec $\mathbb{F}_{2^3}$:

Déterminons avec le même exemple une permutation de $\mathcal{S}_{2^3-1}$ avec le logarithme de Zech. Pour cela, il nous faut un polynôme primitif (dont la racine X est primitive dans $\mathbb{F}_{2^3}$).
On considère le même que précédemment : $X^3 - X - 1$.

On a alors :

$$\begin{array}{lll}
X^{\mathcal{Z}_g(1)} = 1 - X^1 = 1 - X = X + 1 = X^3 & \text{On en déduit} & \sigma(1) = 3 \\
X^{\mathcal{Z}_g(2)} = 1 - X^2 = X^2 + 1 = X^6 & & \sigma(2) = 6 \\
X^{\mathcal{Z}_g(3)} = 1 - X^3 = 1 - (1 + X) = X^1 & & \sigma(3) = 1 \\
X^{\mathcal{Z}_g(4)} = 1 - X^4 = 1 - X^2 - X = X^2 + X + 1 = X^5 & & \sigma(4) = 5 \\
X^{\mathcal{Z}_g(5)} = 1 - X^5 = X^2 + X = X^4 & & \sigma(5) = 4 \\
X^{\mathcal{Z}_g(6)} = 1 - X^6 = X^2 & & \sigma(6) = 2
\end{array}$$

qui a pour matrice de permutation :

$$M(\sigma) = \begin{pmatrix}
- & - & \times & - & - & - \\
- & - & - & - & - & \times \\
\times & - & - & - & - & - \\
- & - & - & - & \times & - \\
- & - & - & \times & - & - \\
- & \times & - & - & - & -
\end{pmatrix}$$

On s'aperçoit bien qu'aucun motif n'est identique, et donc qu'aucune confusion ne peut être faite dans l'analyse du signal de retour.

2.2.2. *Sophistication.*

On peut remarquer en revanche que ce procédé ne fournit qu'une seule permutation parfaite et qui vérifie en plus: $\sigma \circ \sigma = Id$, ce qui peut ne pas être souhaité.

Pour résoudre ces deux problèmes, on fixe une permutation parfaite $\sigma \in \mathcal{S}_{p^\alpha - 2}$ et r un entier premier avec $p^\alpha - 2$. On peut alors générer d'autres permutations parfaites, et non symétriques : on considère $\sigma' \in \mathcal{S}_{p^\alpha - 2}$ définie par :

$$\sigma'(n) := r.\sigma(n)$$

3. PROGRAMMATION

Avertissement

On utilise dans cette partie le logiciel de programmation CAML [3] utilisé en classe préparatoire (voir `http://caml.inria.fr/` pour plus d'information).

Dans ce qui suit, on écrit les polynômes sous forme de vecteurs :

$$[|1,0,1,0,0,1|] = x^5 + x^2 + 1.$$

3.1. FONCTIONS CAML.

On implémente ensuite un certain nombre de fonction intermédiaire, en vue d'écrire la fonction donnant la permutation parfaite et la matrice qui lui est associée :

3.1.1. *Fonction `puiss`.*

```
let rec puiss a b =
    if b=0 then 1
    else a*(puiss a (b-1))
;;
```

La fonction `Puiss` calcule a^b. L'algorithme est peu efficace. On peut implémenter l'exponentiation rapide à la place, mais bon...

Le cardinal du corps de base $\mathbb{F}_p$ est dans ce qui suit 2.

```
let p = 2 ;;
```

3.1.2. *Fonction `sum`.*

```
let rec sum a k b =
    for i = 0 to (vect_length a - 1) do
        b.(k+i) <- (b.(k+i) + a.(i)) mod p
    done;
;;
```

`sum` copie le vecteur `a` en k$^{\text{ième}}$ position dans `b`.
Cela revient à faire $b = b + X^k \times a$.

3.1.3. *Fonction* `scal`.

```
let rec scal a k =
    let b = (copy_vect a) in
        for i = 0 to (vect_length b - 1) do
            b.(i) <- (k * b.(i)) mod p
        done;
b;
;;
```

La fonction `scal` multiplie `a` par le scalaire `k`.

On se donne un polynome irréductible : $x \longmapsto x^5 + x^2 + 1$

```
let poly = [|1;0;1;0;0;1|] ;;
```

On implémente des fonctions liées aux polynômes :

3.1.4. *Fonction* `deg`.

```
let deg = vect_length poly - 1 ;;
```

La fonction deg donne le degré du polynôme.

3.1.5. *Fonction* `resid`.

```
let resid = (scal (sub_vect poly 0 deg) (-1)) ;;
```

`Resid` remplace l'équation $x^5 + x^2 + 1 = 0$ par $x^5 = -x^2 - 1$.

3.1.6. *Fonction* `mult`.

```
let mult a b =
    let c = make_vect ((vect_length a) + (vect_length b)) 0 in
        for i = 0 to (vect_length c - 1) do
            for j = (max (1-(vect_length b)+i) 0)
                        to (min (vect_length a-1) i) do
                c.(i) <- (c.(i) + a.(j) * b.(i-j)) mod p
            done;
        done;
c;
;;
```

La fonction `mult` calcule le polynôme produit (de degré : `deg a + deg b`) de `a` et de `b`.

3.1.7. *Fonction* `mult_F`.

```
let mult_F a b =
    let c = (mult a b) in
        for i = (vect_length c - 1) downto deg do
            sum (scal resid (c.(i))) (i-deg) c
        done;
sub_vect c 0 deg;
;;
```

La fonction `mult_F` fait la multiplication de polynômes dans $\mathbb{F}_{p^{\deg}}$: prend le produit donné par `mult`, et le réduit en utilisant `resid`.

```
mult_F [|0;0;1|] [|0;0;0;1|];;

let l = (puiss p deg - 2);;
```

3.1.8. *Fonction* `equal_F`.

```
let equal_F a b =
    let r = ref true in
        for i = 0 to (deg-1) do
            if ((b.(i) - a.(i)) mod p <> 0) then r := false
        done;
(!r);
;;
```

La fonction `equal_F` permet de savoir si deux polynômes sont égaux.

3.1.9. *Fontion* `tech`.

```
let tech =
    let r = make_vect deg 0 in
        r.(1) <- 1;
r;
;;
```

L'élément X est représenté en vecteur.

3.1.10. *Fonction* `ord_F`.

```
let ord_F =
    let r = make_vect l tech in
        for i = 2 to l do
            r.(i-1) <- (mult_F tech (r.(i-2)))
        done;
r;
;;
```

`ord_F.(i)` est l'élément d'ordre i dans $\mathbb{F}_{p^{\deg}}$.

3.1.11. *Fonction* `unit`.

```
let unit =
    let r = make_vect deg 0 in
        r.(0) <- 1;
r;
;;
```

L'élément 1 représenté en vecteur.

3.1.12. *Fonction* `sum_F`.

```
let rec sum_F a b =
    let r = (copy_vect a) in
        for i = 0 to (deg-1) do
            r.(i) <- (b.(i) + a.(i)) mod p
        done;
r;
;;
```

La fonction `sum_F` fait la somme de deux vecteurs et la réduit modulo p.

```
exception ord of int;;
```

3.1.13. *Fonction* `Zech`.

```
let Zech =
    let r = make_vect l (-1) in
        let find y =
            try
                for i = 1 to l do
        if (equal_F y (ord_F.(i-1))) then raise (ord i)
                done;
                0
                with
                    | ord i -> i in
                            for j = 1 to l do
r.(j-1) <- (find (sum_F unit (scal (ord_F.(j-1)) (-1)))) - 1
                            done;
r;
;;
```

`i` $\mapsto$ `Zech.(i)` est la permutation associée au logarithme de Zech.

3.1.14. *Fonction* `Mat`.

```
let Mat p =
    let r = make_matrix (vect_length p) (vect_length p) "-" in
        for i = 0 to (vect_length p - 1) do
            r.(p.(i)).(i) <- "X"
                done;
r;
;;
```

La fonction `Mat` associe à la permutation fournie par le procédé de Zech, sa représentation matricielle.

3.1.15. *Fonction* `Aff`.

```
let Aff m =
    print_newline();
        for i = 0 to (vect_length m - 1) do
            for j= 0 to (vect_length m.(0) - 1) do
                print_string m.(i).(j);
                print_string " "
            done;
        print_newline();
        done;
;;
```

`Aff` affiche la matrice.

L'entrée de toute ces fonctions donne le retour CAML suivant :

```
**************************************************************************
        puiss :  int -> int -> int = <fun>
        #p :  int = 2
        #sum :  int vect -> int -> int vect -> unit = <fun>
        #scal :  int vect -> int -> int vect = <fun>
        #poly :  int vect = [|1; 0; 1; 0; 0; 1|]
        #deg :  int = 5
        #resid :  int vect = [|-1; 0; -1; 0; 0|]
        #mult :  int vect -> int vect -> int vect = <fun>
        #mult_F : int vect -> int vect -> int vect = <fun>
        #- :  int vect = [|-1; 0; -1; 0; 0|]
        #l :  int = 30
        #equal_F : int vect -> int vect -> bool = <fun>
        #tech :  int vect = [|0; 1; 0; 0; 0|]
        #ord_F : int vect vect = [|[|0; 1; 0; 0; 0|]; [|0; 0; 1; 0; 0|];
[|0; 0; 0; 1; 0|]; [|0; 0; 0; 0; 1|]; [|-1; 0; -1; 0; 0|]; [|0; -1; 0; -1;
0|]; [|0; 0; -1; 0; -1|]; [|1; 0; 1; -1; 0|]; [|0; 1; 0; 1; -1|]; [|1; 0;
0; 0; 1|]; [|-1; 1; -1; 0; 0|]; [|0; -1; 1; -1; 0|]; [|0; 0; -1; 1; -1|];
[|1; 0; 1; -1; 1|]; [|-1; 1; -1; 1; -1|]; [|1; -1; 0; -1; 1|]; [|-1; 1; 0;
0; -1|]; [|1; -1; 0; 0; 0|]; [|0; 1; -1; 0; 0|]; [|0; 0; 1; -1; 0|]; [|0;
0; 0; 1; -1|]; [|1; 0; 1; 0; 1|]; [|-1; 1; -1; 1; 0|]; [|0; -1; 1; -1; 1|];
[|-1; 0; 0; 1; -1|]; [|1; -1; 1; 0; 1|]; [|-1; 1; 0; 1; 0|]; [|0; -1; 1;
0; 1|]; [|-1; 0; 0; 1; 0|]; [|0; -1; 0; 0; 1|]|]
          #unit :  int vect = [|1; 0; 0; 0; 0|]
          #sum_F : int vect -> int vect -> int vect = <fun>
          #Exception ord defined.
          #Zech :  int vect = [|17; 4; 28; 9; 1; 26; 21; 19; 15; 3; 18;
22; 13; 12; 23; 8; 29; 0; 10; 7; 24; 6; 11; 14; 20; 27; 5; 25; 2; 16|]
          #Mat :  int vect -> string vect vect = <fun>
          #Aff :  string vect vect -> unit = <fun>
```

```
*************************************************************************
        Aff ( Mat Zech );;
   - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - -
   - - - - X - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - X -
   - - - - - - - - - X - - - - - - - - - - - - - - - - - - - - - - - - -
   - X - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - X - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - -
   - - - X - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - - - -
   - - - - - - - - X - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - X
   X - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - X - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - X - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - - - - - X - - - - - - -
   - - - - - - X - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - X - - -
   - - - - - X - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - X - - - - -
   - - X - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   - - - - - - - - - - - - - - - - - - - X - - - - - - - - - - - - - - -
        - :  unit = ()
```

4. CONCLUSION

Le problème que rencontrent les radars qui envoient des ondes associées à différentes fréquences peut donc se résoudre. On a vu que la donnée d'un élément primitif dans $\mathbb{F}_p$ suffit à trouver des permutations qui satisfont aux exigences du problèmes. Toutefois, on a montré qu'il était difficile en pratique de connaitre un élément primitif pour p grand. On a donc introduit le logarithme de Zech, qui permet de résoudre le problème en travaillant sur $\mathbb{F}_{p^\alpha}$, en se dotant de polynômes irréductibles. Par ailleurs, on a ensuite trouvé un moyen, lorsqu'on dispose d'une permutation parfaite, de générer plusieurs permutations parfaites différentes les unes des autres, et non symétrique contrairement à celle générée par le logarithme de Zech.

Pour ce qui concerne la faisabilité de la solution, deux remarques peuvent être faites. La première concerne les polynômes irréductibles de $\mathbb{F}_p[X]$: bien qu'ils ne constituent pas le propos de notre TIPE, il est bon de savoir que les polynômes irréductibles de $\mathbb{F}_p[X]$ sont difficiles à obtenir lorsque p et α sont grands, et la recherche générale de ceux-ci, de manière rigoureuse, constitue un problème de grande ampleur. La deuxième remarque concerne les résultats de la programmation : ces derniers montrent que la connaissance d'un polynôme irréductible pour p et α petit, est suffisante pour avoir des applications pratiques satisfaisantes, et qu'il n'est pas nécessaire de savoir théoriquement et rigoureusement comment on obtient un polynôme irréductible quelque soit p et α. Ainsi la matrice donnée par CAML semble de taille largement suffisante.

Pour utiliser le radar de maniére efficace, on pourra alors générer d'autres permutations parfaite à l'aide de cette première permutation et accoler les matrices respectives au fur et à mesure que le temps d'utilisation s'écoule.

References

[1] *Number Theory in Science and Communication*, "Galois Fields", Mr. SCHROEDER.
[2] *Électromagnétisme,* "Métal Parfait Guide d'Onde", Daniel MAURAS
[3] System Release 0.74 ; Institut National de Recherche en Informatique et Automatique (http://www.inria.fr)
[4] On peut trouver une annexe avec quelques explications supplémentaires à la page Internet : http://francoisjaulin.free.fr/Tipe.htm

E-mail address: francoisjaulin@yahoo.fr
URL: http://francoisjaulin.free.fr/Tipe.htm