Rapport Final PROTIS

Transmission cohérente

Groupe 1 :
Xiaohui Zhao
Mathias El Baz

Institut d'Optique Graduate School, Palaiseau
En partenariat avec Nokia Bell Labs
Encadré par Sylvain Almonacil, Nicolas Dubreuil et Julien Villemejane

Octobre 2019 - Mai 2020

Table des matières

Introduction

Dans un monde où transporter chaque jour plus d'information plus rapidement est ce qui permet de maintenir notre niveau de vie en répondant à nos besoins journaliers, l'amélioration de nos moyens de télécommunication devient un enjeu majeur de la société. 700 milliards de minutes sont passées chaque mois sur Facebook au niveau mondial. 50 millions de tweets sont diffusés quotidiennement et Google traite 24 pétaoctets de données par jour transportées le long de fibres optiques.

Deux solutions viennent à l'esprit afin d'augmenter la quantité d'information transmise : augmenter le nombre de fibres entre les noeuds du réseau ou bien augmenter la capacité de transmission de chaque fibre. La deuxième solution semble à la fois plus viable économiquement, techniquement et environnnementalement.

En 2012, l'idée d'utiliser les deux polarisations du champ électrique transmis est émise. De cette méthode, on pourrait donc doubler le volume d'information horaire d'une fibre. De plus, les deux polarisations risquent de peu interférer le long de la fibre car étant toute deux orthogonales à l'origine. Ainsi, contrairement à un ajout de mode, le rapport signal sur bruit du signal reçu ne devarit théoriquement pas changer. Néanmoins, cette technique nécessite un traitement de l'information à l'arrivée encore plus lourd, afin notamment de pouvoir retrouver les polarisations d'origine. Le projet de PIMS a débuté en 2013 et nous avons voulu continuer cette aventure lors de cette édition 2019-2020.

En plus des heures normalement consacrées au PIMS, nous nous sommes penchés sur le sujet dix-huit heures de plus dans le cadre de PROTIS. Cette période fut l'occasion de revenir sur le traitement numérique du signal à la réception, de comprendre dans les détails chaque algorithme et d'en proposer une version moins gourmande de place et plus rapide, chose nécessaire si la rapidité de l'acheminement du signal est un enjeu majeur des télécommunications.

1 Présentation du système

L'objectif du projet est de transmettre de l'information via un signal lumineux le long de N*200km de fibre optique en utilisant un sytème dit de transmission cohérente. On distingue trois sous-systèmes au sein du montage : la chaîne d'émission, la boucle de recirculation et la partie réception.

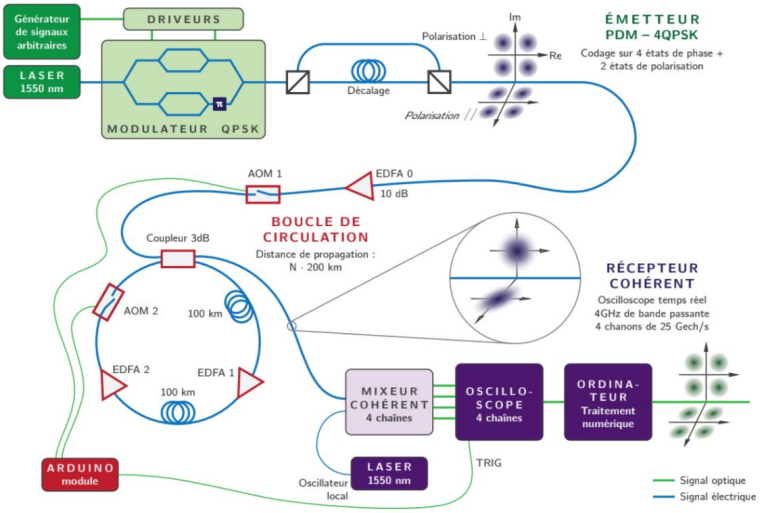

FIGURE 1 – Schéma du montage de transmission cohérente

1.1 L'émission

L'émission consiste à injecter dans la fibre un signal lumineux où l'on aura encodé l'information que l'on désire transmettre sur chacune des polarisations.Les deux polarisations du faisceau laser à 1550nm sont chacune envoyées dans un modulateur Mach Zender qui va encoder l'information avec une modulation QPSK (modulation en phase). On transmet donc 2 bits d'informations par symbole que l'on envoie à 5 GBaud via un générateur de séquences pseudo-aléatoire pour chaque polarisation. On transmet donc 20 Gbit d'information par seconde. Puisque l'on ne dispose pas de deux générateurs différents, on applique la même séquence sur les deux polarisations que l'on déphase l'une par rapport à l'autre ensuite au moyen d'une fibre à retard.

Tâchons cependant dans une courte apparté d'expliquer un peu plus en détails le fonctionnement d'un modulateur Mach Zenhder (MZM). Il s'agit d'un interféromètre constitué de deux coupleurs 3dB qui sépare puis recombine deux parties d'énergie égale du faisceau lumineux. Sur chaque bras, le faisceau rencontre un modulateur électro- soumis à une tension V_i qui déphase en conséquence le faisceau. Le champ E_{out} s'écrit :

$$E_{out} \propto E_{in} \cos(\pi \frac{V_2 - V_1}{V_\pi}) \tag{1}$$

où V_π dépend du modulateur. Avec une différence de tension nulle, on obtient le point de phase nulle et au contraire une différence de tension de V_π permet d'obtenir le point opposé de la constellation QPSK.

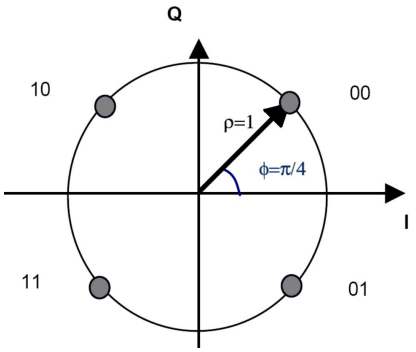

FIGURE 2 – Signal QPSK

Pour obtenir des formats de modulation plus complexes, on utilise deux MZM montés sur chaque bras d'un troisième MZM. Un des bras est déphasé de $\pi/2$ Cela nous permet de parcourir tout le plan complexe en phase du champ optique. On peut ainsi coder de l'information sur la partie reelle et la partie imaginaire du champ optique. On obtient ainsi une constellation de quatre points en sortie de même module mais de phase différente.

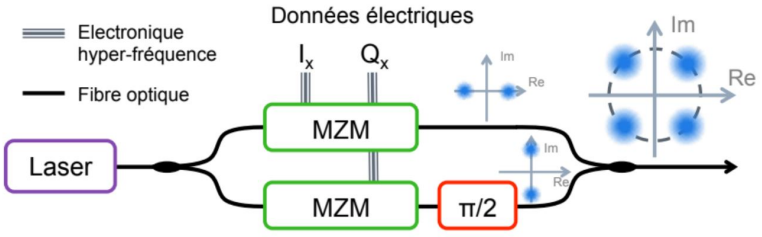

FIGURE 3 – Double MZM pour générer la constellation QPSK

Enfin, la modulation sur deux polarisations en QPSK, ou appelée PDM-QPSK, nécessite de séparer les deux polarisations puis de les retarder l'un par rapport à l'autre avec une fibre à retard, afin de créer un signal différent sur chacune des polarisations. Le tout est assimilable au schéma ci-dessous avec deux générateurs de séquences :

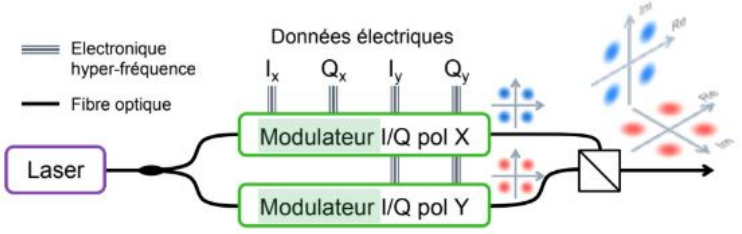

FIGURE 4 – Schéma du système de modulation

1.2 La boucle

L'un des défis du projet est de pouvoir transmettre le signal sur des milliers de kilomètres afin de modéliser les grands réseaux de télécommunication actuels. Une solution trouvée et conçue par l'équipe précédente fut de faire circuler le signal dans une boucle de 200km. Ainsi, si N est le nombre de tours, la lumière parcoura N*200km.A noter qu'un programme pour faire marcher la boucle a été rédigé sur l'ordinateur, il faut chercher le code example.m.

Pour ce faire, on injecte un signal pendant le temps de remplissage de la boucle (1 ms pour une boucle de 200km). La commande est faite à partir de deux interrupteurs acousto-optiques (AOM) contrôlés par des boitiés où l'utilisateur a seulement à saisir le nombre de tours dans la boucle et la durée de remplissage. Les AOM permettent d'injecter puis de faire ressortir le signal en passant par un coupleur -3dB. Les AOMs sont gérés par des drivers, qui sont branchés à des alimentations de 24 V et de 5 V.

De plus, l'atténuation de 0.2 dB/km diminue la puissance du signal d'un facteur 100 chaque 100 kilomètres. Pour compenser ces pertes, on utilise des amplificateurs à fibre dopée erbium (EDFA) que l'on dispose chaque 100 km : un au milieu de boucle, l'autre à la fin. Le signal est également amplifié une seule fois en entrée de boucle par un EDFA0.

Dans un premier temps, on pourra tester une transmission en Back-to-Back, c'est à dire ne pas utiliser la boucle et relier directement la chaîne d'émission à celle de réception. On pourra ainsi tester la DSP avec des signaux peu bruités.

Voici un schéma de la boucle pour plus de clarté :

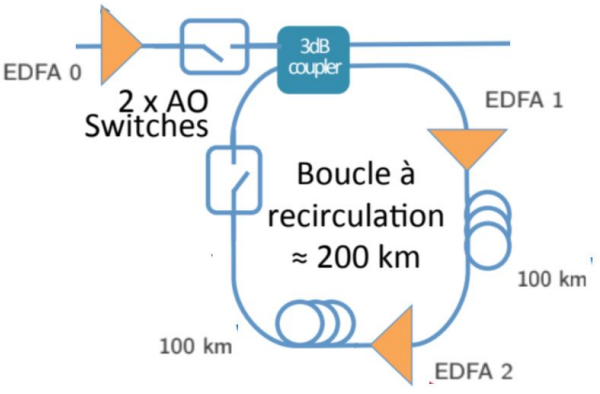

FIGURE 5 – Boucle de recirculation

1.3 La réception cohérente

Le circuit de réception cohérente fait toute la particularité du projet.Les deux polarisations complexes sortent de la boucle dans un seul faisceau lumineux ; l'objectif est donc de retrouver à partir de ce faisceau quatre données correspondant aux parties imaginaires et réelles des deux polarisations.

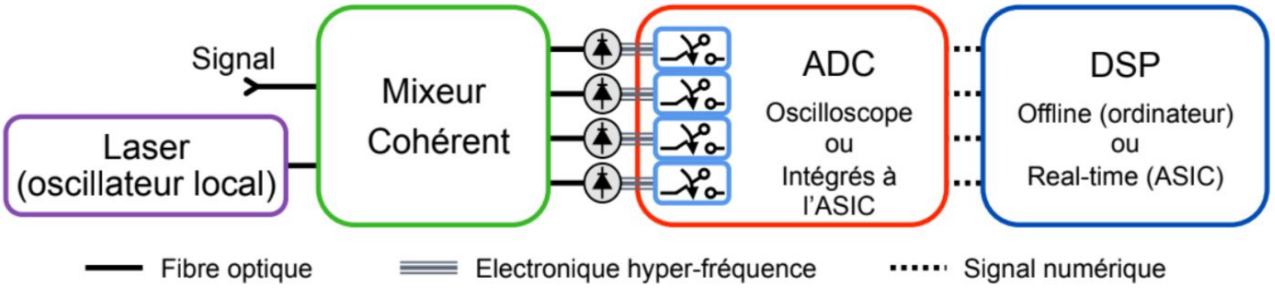

FIGURE 6 – Schéma du sysème de réception cohérente

La réception cohérente repose sur l'utilisation d'un mixeur cohérent (Attention ! Allumer les photodiodes avant d'injecter le signal car le mixeur est très fragile et onéreux).

Une photodiode ne peut mesurer des variation de phase. Notre modulation ne serait donc pas visible si l'on ne convertit pas nos variations de phase en variations d'intensité lumineuse. C'est ce que permet le principe d'interférence ! On fait interférer le laser arrivant avec un oscillateur local. Le schéma ci-dessus montre les principaux composants qui constituent le récepteur cohérent. Le signal va battre avec un laser de même longueur d'onde que celle du laser d' emission. On conseille de lire la thèse de Philippe Jennevé pour plus de détails.

Grossièrement, le mixeur cohérent donne huit sorties connectées à quatre photodiodes balancées permettant de récupérer les quatre informations de champs dont nous avons besoin pour retrouver les deux champs selon les deux polarisations. Les courants des photodiodes sont alors échantillonés par un convertisseur analogique-numérique vus en temps réel sur un oscilloscope Dans un premier temps, on pourra effectuer une transmission homodyne (en opposion à hétérodyne), c'est à dire qu'on utilise aussi le signal comme oscillateur local afin de ne pas se soucier du désaccord entre les deux lasers lors de la DSP, notamment en ce qui concerne le désaccord de fréquence.

S'ensuit alors la partie de traitement du signal pour retrouver l'information qui a été altérée lors du trajet.

2 Digital signal processing (DSP)

Notre procédé de télécommunication recquiert un traitement du signal reçu très lourd. Si l'on s'amuse à afficher les deux champs complexes associés au deux polarisations sur l'ordinateur, la constellation n'est pas présente. L'amplitude n'est pas constante, on obtient un disque. De nombreux phénomènes dégradent notre signal comme la dispersion chromatique ou le fait que l'orientation des polarisations change avec les torsions de la fibre. Seul l'application de plusieurs algorithmes successifs permet de retrouver l'information transmise.

Avant toute démarche de DSP, il est important de reconstruire les deux champs complexes associés aux deux polarisations en sommant la partie rélle et (i fois) la partie imaginaire.

2.1 Algorithme de rééchantillonnage

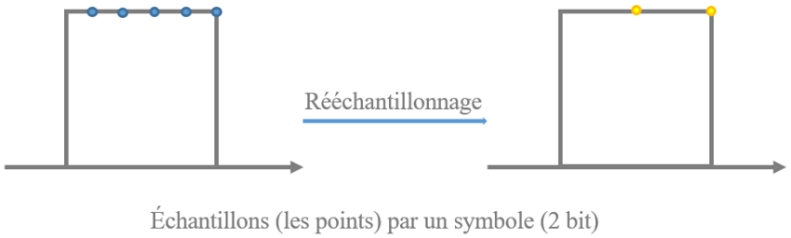

FIGURE 7 – Schéma du rééchantillonnage

Pour traiter le signal correctement, nous n'avons besoin que de 2 échantillons par symboles (2 sps).En effet, nous verrons plus tard que deux échantillons sont nécessaires car nous devrons n'en garder qu'un sur deux après le Constant Modulus Algorithm. Or l'oscilloscope échantillonne souvent à une fréquence souvent différente de deux fois le débit symbole. On effectue donc au préalable un rééchantillonage du signal en "contractant" ou "dilatant" le signal.

Le rééchantillonnage se fait très simplement via la fonction resample de Matlab qui prend trois arguments, un champ et deux entiers p et q.Le facteur de rééchantillonage du signal sera p/q. Ainsi si l'on veut deux sps, p sera égal à 2sps*débit de symbole et q à la fréquence d'échantillonage de l'oscilloscope.

Le signal obtenu est donné ci-dessus. La constellation n'est pas encore visible, il nous faudra appliquer différents algoritmes afin de retrouver l'amplitude et la phase de la constellation QPSK.

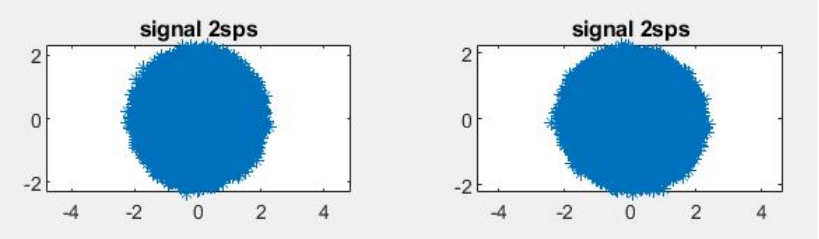

FIGURE 8 – Champs complexes rééchantilloné à 2sps des deux polarisations

2.2 Filtrage pour compenser la dispersion chromatique

L'indice d'un milieu vu par une onde lumineuse change selon sa longueur d'onde, ce qui se traduit par un élargissement des impulsions envoyées dans la fibre. Dans la fibre optique, l'enveloppe du champ suit l'équation :

$$\frac{\partial A(z,t)}{\partial z} = j\frac{D\lambda^2}{4\pi c}\frac{\partial^2 A(z,t))}{\partial t^2} \tag{2}$$

avec D le facteur de dispersion, souvent en ps/(nm*km). Dans notre cas, D=17 ps/(nm*km), c'est à dire que deux longueur d'onde espacée d'un nanomètre auront un retard relatif de 17ps après un kilomètre parcouru.

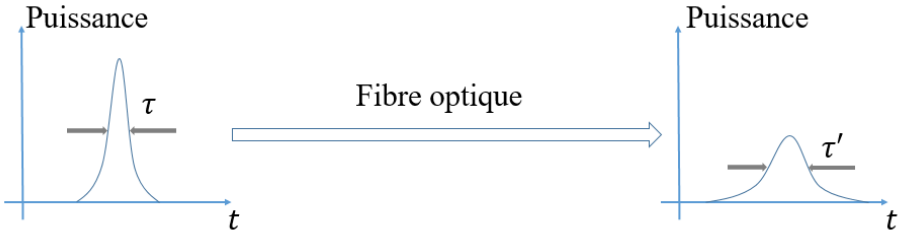

FIGURE 9 – Etalement temporel des impulsions à cause de la dispersation chromatique

Pour compenser cette dispersion, on applique son filtre inverse au champ, c'est à dire qu'on multiplie la transformée de Fourier du signal par une fonction de transfert de la forme :

$$H(\omega) = \exp(i\frac{\beta_2 L}{2}\omega^2) \tag{3}$$

avec L la longueur parcourue par le faisceau lumineux dans la fibre.

Le signal reste le même sans distinction d'amplitude ou de phase. Il nous faudra faire appel au CMA pour retrouver l'amplitude de notre modulation QPSK...

2.3 Constant Modulus Algorithm (CMA)

L'ampliude non constante du signal en sortie est intimement lié à la rotation du champ optique à l'intérieur de la fibre et au retard que prend une polarisation par rapport à l'autre. En effet, comment peut on être sûr que les axes des deux polarisations n'aient pas changé et donc que les deux polarisations observées sur l'ordinateur ne soient pas des combinaisons linéaires des deux polarisations à l'origine ?

On note $E_r = \begin{cases} E_r^x \\ E_r^y \end{cases}$ le champ à l'entrée du mixeur cohérent et $E_t = \begin{cases} E_t^x \\ E_t^y \end{cases}$ le champ transmis. En supposant que le champ a seulement tourné d'un angle θ inconnu, on a la relation :

$$E_r = \mathbf{R}E_t = \begin{pmatrix} \cos\theta & \exp(-j\phi)\sin\theta \\ -\exp(j\phi)\sin\theta & \cos\theta \end{pmatrix} E_t \tag{4}$$

Tout l'enjeu de l'algorithme est de retrouver cette matrice R, i.e, de retrouver la réponse impulsionnelle de la fibre optique, ou matrice de Jones. Ceci permettra de séparer les deux polarisations et récupérer le signal d'entrée. L'algorithme de module constant a pour fonction de minimiser un critère d'erreur $\varepsilon^2 = \begin{cases} \varepsilon_x^2 = 1 - \mid E_x \mid^2 \\ \varepsilon_y^2 = 1 - \mid E_y \mid^2 \end{cases}$, ce qui équivaut à dire que l'algorithme cherchera à obtenir deux polarisations de module constant égal à un.

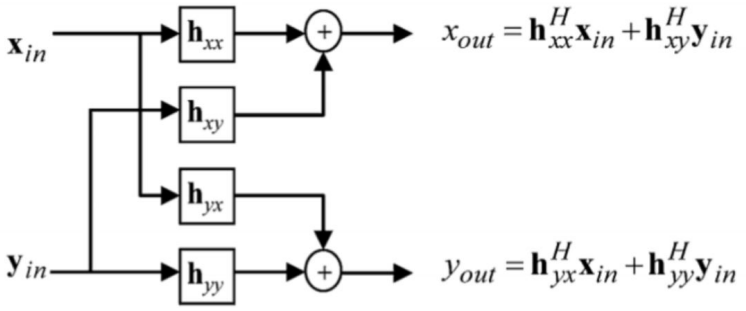

FIGURE 10 – Représentation de l'opération de filtrage de la CMA

Il y a cependant une chose que l'on a mis sous silence et qui va nous rendre l'écriture de l'algorithme plus compliquée. Nous avons supposé plus haut que la matrice **R** était la même pour tous les échantillons, ce qui n'est pas envisageable si chaque polarisation ne va pas à la même vitesse dans la fibre. Nous remarquons en effet qu'appliquer le même filtre à tous les échantillons ne permet pas d'aboutir à un résultat satisfaisant. Ainsi le filtre doit être mis à jour au fur et à mesure des échantillons.

Le filtre est appelé MIMO, c'est à dire multi-entrées, multi-sorties. En effet, chaque polarisation de sortie dépend des deux polarisations d'entrée comme on peut le voir ci-dessous. Les quatre coordonnées du filtre h_{xx}, h_{xy}, h_{yx} et h_{yy} dépandent de l'échantillon.
Pour avancer un peu plus dans la difficulté, la dispersion de polarisation dépend elle-même de la fréquence, autrement dit, la vitesse d'une polarisation l'une par rapport à l'autre n'est pas la même dans le temps car n'étant pas la même pour chaque longueur d'onde. On ne peut donc pas retrouver les polarisations d'entée avec un seul échantillon et le filtre devra prendre en compte les valeurs de champ sur N échantillons, pendant un temps qui doit environ couvrir un décalage typique d'une polarisation par rapport à une autre. Cette valeur N (aussi appelée taps) correspond donc à la longueur du filtre h. On applique donc au signal reçu le filtrage :

$$x_{\text{out}}[k] = \mathbf{h}_{xx}^H[k]\mathbf{x}_{\text{in}}[k] + \mathbf{h}_{xy}^H[k]\mathbf{y}_{\text{in}}[k]$$
$$y_{\text{out}}[k] = \mathbf{h}_{yx}^H[k]\mathbf{x}_{\text{in}}[k] + \mathbf{h}_{yy}^H[k]\mathbf{y}_{\text{in}}[k]$$

FIGURE 11 – Application du filtre au signal

où $x_{in} = [x_{in}(k)...x_{in}(k-N-1)]$.

L'algorithme va chercher à minimiser $\varepsilon_x^2 = (1 - |x_{out}|^2)$, autrement dit l'écart de la norme du vecteur de sortie à la norme unitaire (de même pour le vecteur de polarisation y). On choisit un filtre h initial de telle sorte que pour tout k de [1 :N] :

$$\begin{aligned}
\mathbf{h}_{xx} &= \mathbf{h}_{xx} + \mu\varepsilon_x \mathbf{x_{in}} x^*_{\text{out}} \\
\mathbf{h}_{xy} &= \mathbf{h}_{xy} + \mu\varepsilon_x \mathbf{y_{in}} x^*_{\text{out}} \\
\mathbf{h}_{yx} &= \mathbf{h}_{yx} + \mu\varepsilon_y \mathbf{x_{in}} y^*_{\text{out}} \\
\mathbf{h}_{yy} &= \mathbf{h}_{yy} + \mu\varepsilon_y \mathbf{y_{in}} y^*_{\text{out}}
\end{aligned}$$

FIGURE 12 – Réactualisation du filtre

avec μ le facteur de convergence. La longueur N du filtre est usuellement très faible devant la taille des vecteurs des signaux complexes que l'on reçoit. Notamment, dans notre cas, les vecteurs possèdent 200 000 échantillons. Le filtre est donc appliqué sur une toute petite partie du vecteur. C'est un filtre glissant.

On l'applique successivement à des parties de taille N des deux vecteurs en sortie de l'oscilloscope en le faisant converger à chaque fois. Converger c'est-à-dire, en cherchant à minimiser l'erreur, en mettant à jour le filtre comme expliqué précédemment.

On choisit N = 15, appelé taps dans le code. Cette valeur doit être impaire afin d'avoir un filtre symétrique autour de la valeur centrale. Cette valeur est choisie puisqu'elle correspond, en ordre de grandeur, au décalage typique d'une polarisation par rapport à une autre. On initialise h à l'identité.

Point important : on ne met à jour le filtre que sur un échantillon sur deux et pareil pour l'estimation de l'erreur. On ne peut effectivement pas forcer tous les échantillons à être sur le cercle unité, puisque l'on est encore à deux échantillons par symbole. Ainsi, l'algorithme revient grossièrement à forcer un échantillon sur deux à se placer sur le cercle unité. Cet échantillon va être construit comme un mélange de N échantillons (situés de part et d'autre dans le temps) du signal sortant de l'oscilloscope.

On choisit un facteur de convergence d'environ 10^{-3}. Plus le facteur est petit, meilleure est la précision mais l'algorithme est plus lent à converger.

Les résultats sont donnés ci-dessous pour une trace expérimentale de 200 000 échantillons.

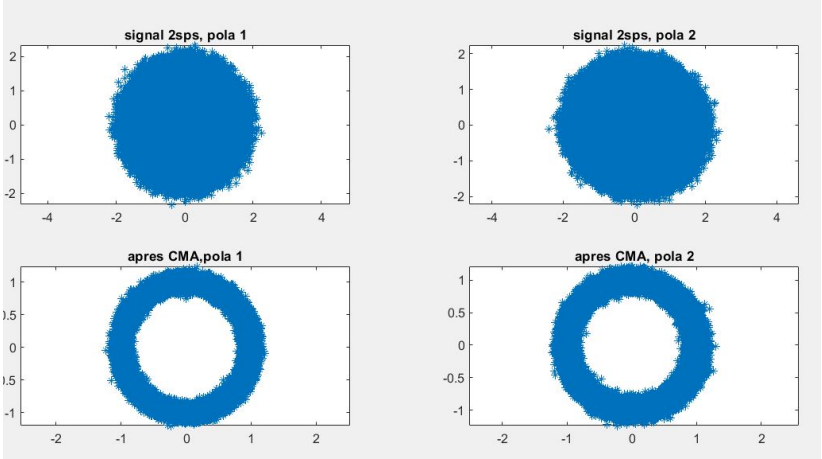

FIGURE 13 – Résultat de l'application du CMA

Pour le graphique de la fonction d'erreur, on a préféré moyenner par paquet de 100 symboles afin de le rendre plus visible.

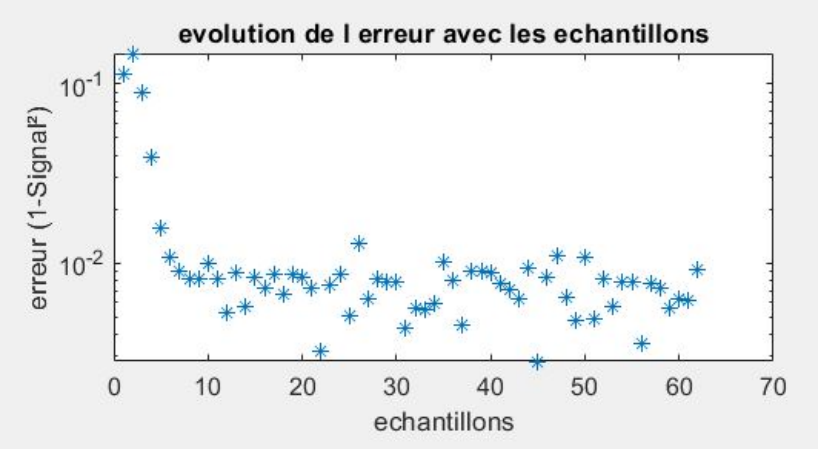

FIGURE 14 – Evolution de la fonction d'erreur avec les symboles

2.4 Algorithme d'estimation de l'erreur de fréquence

On fait interférer dans le mixeur cohérent notre signal laser avec un oscillateur local (ie un autre laser). Néanmoins, nous ne sommes pas certain que ces deux lasers aient la même fréquence, premièrement parce qu'ils ne sont pas exactement accordés et que la fréquence centrale du signal laser a pu être modifié lors de la transmission.

De la formule des interférences, on en déduit que le champ complexe tournera à $\Delta\omega(t)$, le désaccord de pulsation au temps t. Le but de cet algorithme sera donc de retrouver ce désaccord (pour chaque valeur de signal !) et de multiplier le champ complexe dans le domaine temporel par $e^{-i\Delta\omega(t)t}$.

La valeur du champ lors du nième échantillon s'écrit :

$$x(n) = \exp(i(\theta(n) + 2\pi n \Delta f_n T_{symb})) \tag{5}$$

où T_{symb} est la durée d'un symbole et $\theta(n)$ est la phase d'origine du point, multiple donc de $\frac{\pi}{4}$.

Un bon estimateur pour Δf_n est donné itérativement par :

$$\Delta f_n = \Delta f_{n-1}(1-\mu) + \mu \arg(\frac{(x_n x_{n-1}^*)^4}{8\pi T_{symb}}) \tag{6}$$

où μ est un paramètre de convergence qui détermine la sensibilité de la méthode. Plus μ est petit plus l'algorithme ira par petits accoups, et donc plus l'algorithme sera plus sensible mais plus lent. Ce paramètre est généralement de l'ordre de 0,0001 à 0,01.

Avec la puissance 4, on annule l'argument lié à la phase d'origine.

Il suffit alors de multiplier pour chaque polarisation le champ complexe par la matrice des $e^{-2i\pi n \Delta f_n T_{symb}}$ afin de plus ou moins stopper la rotation des points sur le cercle unité.

Voici une simulation de l'évolution de l'estimation pour $\mu = 0,005$ et une erreur de fréquence constante de 1 GHz.

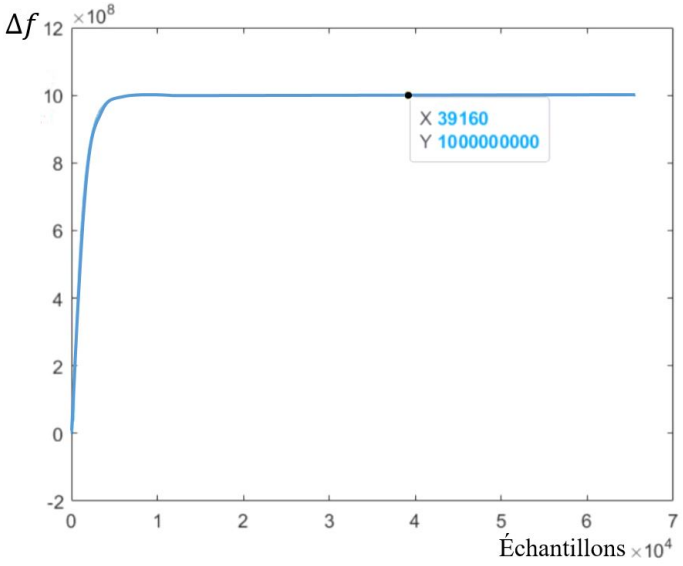

FIGURE 15 – Rapidité de la convergence de l'algorithme d'estimation de fréquence

2.5 Carrier Phase Estimation (CPE)

On cherche à estimer la phase de la porteuse pour compenser les variations de phase liées à la détection hétérodyne (entre autres). Dans notre cas du QPSK, l'algorithme de Viterbi est bien adapté. On compense la phase en soustrayant la phase résiduelle obtenue en prenant la puissance 4 du champ reçu. L'estimation de la phase joue le rôle d'un filtre passe haut. Il faut choisir alors la longueur du filtre qui va modifier la fréquence de coupure du filtre. Par exemple, dans le cas de variations lentes de phase, un filtre de grande longueur sera plus adapté.
Rapide explication : En sortie de la fibre, le champ complexe est de la forme $e^{i(\frac{\pi}{4}+k\frac{\pi}{2}+\varphi)}$ avec φ la phase inconnue qui s'est ajoutée lors de la transmission.

Si on prend en puissance 4 ce champ, on on peut déduire directement la valeur de φ en cherchant l'argument. On multiplie alors le champ initial par $e^{-i\varphi}$ et le tour est joué !

L'algorithme est choisi est donc très proche du précédent, si ce n'est que le filtre appliqué est maintenant vectoriel de la dimension du taps. On passe donc d'un filtre de dimension 1 avec l'estimateur de fréquence à un filtre de dimension N>1, afin de corriger les perturbations plus lentes de variations de phase. Pour notre montage, on a trouvé empiriquement qu'une valeur de taps N=11 était optimale pour retrouver les 4 points de la constellation QPSK.

Il suffit alors d'associer à chaque symbole son couple de bit associé, le signal a été décodé !

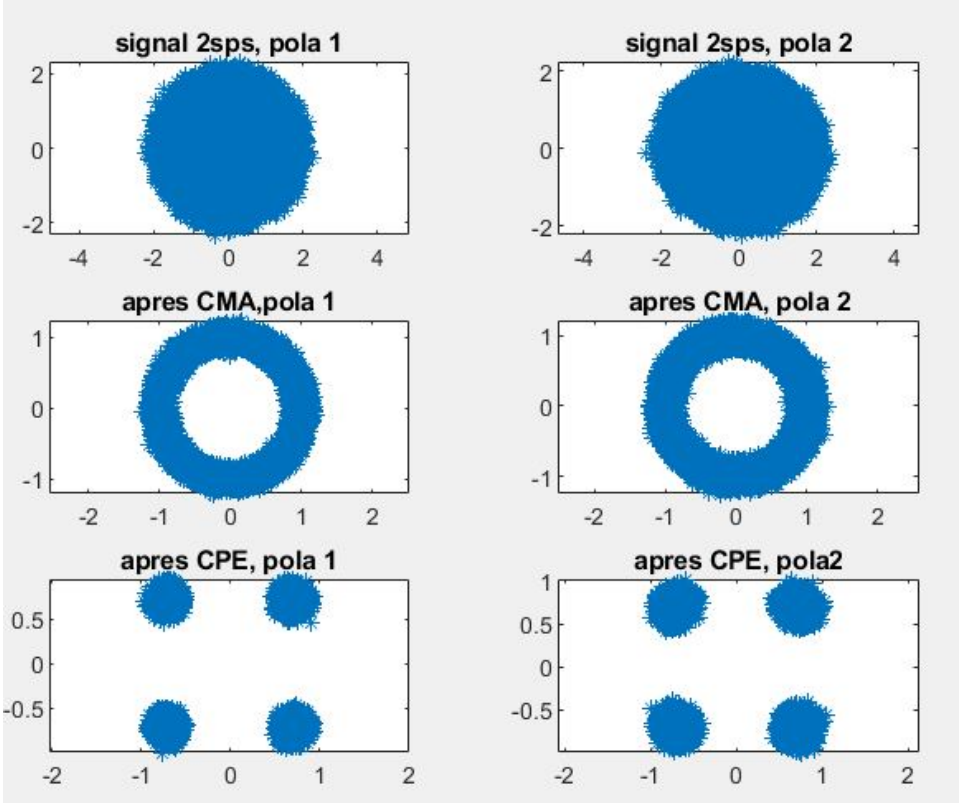

FIGURE 16 – De la deuxième ligne à la troisième : application du CPE

3 Codes de la DSP

3.1 Main

```
clear all

%% Definitions
% Normalization function handle
Pnormm = @(x) (x - mean(x))./sqrt(mean(abs((x - mean(x)).^2)));

% DSP parameters configuration
p.c=3*1e8;% Velocity of the light (in m/s)
p.wavelength= 1.55*1e-6;% Wavelength of the laser (m)
p.scope_sampling_rate = 40*1e6;% Sampling rate of the scope
p.symbol_rate = 32*1e6 % Symbol rate in Baud
p.dsp_sps = 2; % Targeted number of SPS in DSP
p.distance_fiber = 0; % Distance traveled in the fiber (m)
p.dispersion= 1.7*1e-5;% Dispersion for 1550nm (in s/m )
p.beta2 = -p.wavelength^2*p.dispersion/(2*pi*p.c);% beta2
    cefficient (in ISU)


%% Here starts DSP algorithms
% Loading field from a file
load('trace_experimentale');

%Construction du champ
n_fft = 200000;
fieldc=zeros(n_fft,2);
fieldc(1:n_fft,1)=field(1:n_fft,1)+1i*field(1:n_fft,2);
fieldc(1:n_fft,2)=field(1:n_fft,3)+1i*field(1:n_fft,4);

time_vect = (0:1/p.scope_sampling_rate:((n_fft-1)/p.
    scope_sampling_rate))';
frequency_vect = ((-p.scope_sampling_rate/2+p.
    scope_sampling_rate/n_fft):p.scope_sampling_rate/n_fft:(p.
    scope_sampling_rate/2))';
figure;
plot(frequency_vect, 10.*log10(abs(fftshift(fft(fieldc)))))

% Normalization after the scope
fieldn = Pnormm(fieldc);

% Resampling
```

```
field_2sps = Pnormm(resample(fieldn, p.symbol_rate*p.dsp_sps, p.scope_sampling_rate));
% Computing time and frequency vector after resampling
n_fft = size(field_2sps,1);
frequency_vect = ((-p.symbol_rate*p.dsp_sps/2+p.symbol_rate*p.dsp_sps/n_fft):p.symbol_rate*p.dsp_sps/n_fft:(p.symbol_rate*p.dsp_sps/2))';

% Dispersion compensation
% Define the CD operator
cd_operator = exp(+1i*2*pi^2*p.beta2*p.distance_fiber.*frequency_vect.^2);
% Apply the CD operator
fielddc = ifft(fftshift(cd_operator.*fftshift(fft(field_2sps))));
fielddc = Pnormm(fielddc);

% CMA
field_cma=NewCMA(fielddc);
field_cma = Pnormm(field_cma);
plot(abs(field_cma(:,1)))

%Frequency estimation and correction
time_vect2=(1:size(field_cma))'/ p.symbol_rate;

[df1,df2] = cfe(field_cma,0.001,1/(p.symbol_rate ));
field_cfe=field_cma;
field_cfe(:,1)=field_cfe(:,1).*exp(-1i*2*pi*df1.*time_vect2);
field_cfe(:,2)=field_cfe(:,2).*exp(-1i*2*pi*df2.*time_vect2);

% Phase compensation
field_cpe=phase_compensation(field_cma);
field_cpe=Pnormm(field_cpe);

%Plot the different steps of the DSP
figure
subplot(3,2,1)
plot(field_2sps(:,1),'*')
title('signal 2sps, pola 1')
axis equal
subplot(3,2,2)
plot(field_2sps(:,2),'*')
title('signal 2sps, pola 2')
```

```
75 axis equal
76 subplot(3,2,3)
77 plot(field_cma(30000:2:end-30,1),'*')
78 title('apres CMA, pola 1')
79 axis equal
80 subplot(3,2,4)
81 plot(field_cma(30000:2:end-30,2),'*')
82 title('apres CMA, pola 2')
83 axis equal
84 subplot(3,2,5)
85 plot(field_cpe(20000:2:end-30,1)*exp(1i*pi/4),'*')
86 title('apres CPE, pola 1')
87 axis equal
88 subplot(3,2,6)
89 plot(field_cpe(20000:2:end-30,2)*exp(1i*pi/4),'*')
90 title('apres CPE, pola2')
91 axis equal
```

3.2 CMA

```
function  field_out=NewCMA(field_in)
% Normalisation
Pnormm = @(x) (x - mean(x))./sqrt(mean(abs((x - mean(x)).^2)));
fields_in = Pnormm(field_in); %Normalisation du champ

% param tre du CMA
theta = 0;
phi = 0;
taps = 31; % longueur N du filtre , entier impair pour avoir un
    filtre sym trique autour du tap central
mu = 1e-3; % plus la correction est petite , meilleure est la
    pr cision de convergence mais elle est plus lente et
    inversement si mu est plus grand (typique 1e-2 / 1e-4).
n = size(field_in,1);

% initialisation du filtre h
ho = [cos(theta) exp(-1i*phi)*sin(theta) ; -exp(1i*phi)*sin(
    theta) cos(theta)]; %Matrice 2*2 Identit
h =1j* zeros(taps,2,2); %On construit un filtre de longueur N=
    taps
h((taps+1)/2,:,:) = ho; %Filtre h nul sauf au milieu ou c'est l'
    identit
%h(k,i,j) avec 1=<k=<N et 1=<i,j=<2

field_beta=zeros(n,2);
if size(field_in(1,:))==4
     field_beta(:,1)=field_in(:,1)+1i*field_in(:,2);
     field_beta(:,2)=field_in(:,3)+1i*field_in(:,4);
else
     field_beta=field_in;
end

field_alpha=field_beta;

%tableaux qui contiendront les erreurs (1-champ^2)
epsx2_rec=1j*ones(floor((n-taps-1)/2),1);%on garde une trace des
    erreurs
epsy2_rec=1j*ones(floor((n-taps-1)/2),1);

for ii = 1:(n-taps-1) % boucle temporelle sur les    chantillons

     for polout = 1:2 % boucle sur les polarisations de sorties
```

```
36          % application directe du filtre h sur le champ d'entr e
37          field_beta(ii,polout)= (h(:,polout,1).')*field_alpha(ii:
                ii+taps-1,1) + (h(:,polout,2).')*field_alpha(ii:ii+
                taps-1,2);
38      end
39
40      if mod(ii,2) == 0 % mise a jour du filtre h un    chantillon
            sur deux
41
42          epsx2 = mu*(1-abs(field_beta(ii,1))^2); % erreur de
                convergence x le coeff de correction mu
43          epsy2 = mu*(1-abs(field_beta(ii,2))^2);
44
45          epsx2_rec(ii/2) = epsx2; % juste pour sauvegarder les
                erreurs
46          epsy2_rec(ii/2) = epsy2;
47
48          % mise a jour du filtre h en fonction de l'erreur (via
                epsx2)
49          h(:,1,1) = h(:,1,1) + (epsx2*conj(field_alpha(ii:ii+taps
                -1,1))*field_beta(ii,1));
50          h(:,1,2) = h(:,1,2) + (epsx2*conj(field_alpha(ii:ii+taps
                -1,2))*field_beta(ii,1));
51          h(:,2,1) = h(:,2,1) + (epsy2*conj(field_alpha(ii:ii+taps
                -1,1))*field_beta(ii,2));
52          h(:,2,2) = h(:,2,2) + (epsy2*conj(field_alpha(ii:ii+taps
                -1,2))*field_beta(ii,2));
53
54      end
55  end
56
57  n=size(field_beta,1)
58
59  %Ecriture de fieldout, on ne garde qu'un    chantillon    sur deux (
        on passe    1sps
60  field_out=zeros(int32(n/2),2);
61  for i=2:2:n
62      field_out(fix((i)/2),1)=field_beta(i,1);
63      field_out(fix((i)/2),2)=field_beta(i,2);
64  end
65
66  %trac  de l'erreur pour des moyennes de paquets de 500 symboles
67  y=1:62;
```

```
for i=1:62
    y(i)=0;
    for j=1:500
        y(i)=epsx2_rec((i-1)*500+j)+y(i);
    end
end
semilogy(y,'*')
title('evolution de l erreur avec les echantillons')
xlabel('echantillons')
ylabel('erreur (1-Signal^2)')
```

3.3 CFE (Carrier frequency estimation)

```
function [df1,df2] = cfe(field_cma,mu,T)%dfi tableau des
    desaccords pour la polarisation i, field_cma champs en sortie
     de CMA, mu le facteur de convergence, T la p riode d'un
    symbole
n=size(field_cma, 1);
df1=zeros(n,1);
df2=zeros(n,1);
%calcul des desaccords avec l'algorithme de Vitterbi
for i =2: n
df1(i)=df1(i-1)*(1-mu)+mu*angle((field_cma(i,1)*conj(field_cma(i
    -1,1)))^4)/(8*pi*T) ;
df2(i)=df2(i-1)*(1-mu)+mu*angle((field_cma(i,2)*conj(field_cma(i
    -1,2)))^4)/(8*pi*T) ;
end
```

3.4 CPE (Carrier phase estimation)

```
function field_phase=phase_compensation(field_CFE)%field_CFE est
    le signal en sortie de CFE
[data_length,~]=size(field_CFE);
field_CMA4=field_CFE.^4;%Elimine la phase de modulation
angle_field4=angle(field_CMA4);
angle_averageX4=mean(angle_field4(:,1));
angle_averageY4=mean(angle_field4(:,2));
angle_averageX=angle_averageX4/4;
angle_averageY=angle_averageY4/4;
field_rotate=[field_CFE(:,1)*exp(-1j*angle_averageX)  field_CFE
   (:,2)*exp(-1j*angle_averageY)];
Tap_number=11;%longueur du filtre
angle_correction=zeros(data_length,2);%Angle utilis  pour
   corriger la phase
%Calcul de cet angle
for ii=1:data_length-Tap_number+1
    angle_tapX=unwrap(angle_field4(ii:ii+Tap_number-1,1))/4;
    angle_tapY=unwrap(angle(field_rotate(ii:ii+Tap_number-1,2)
       .^4))/4;
    angle_tapX_ave=mean(angle_tapX);
    angle_tapY_ave=mean(angle_tapY);
    angle_correction(ii+(Tap_number-1)/2,1)=angle_tapX_ave;
    angle_correction(ii+(Tap_number-1)/2,2)=angle_tapY_ave;
end
coeff_correction=exp(-1j*angle_correction);
field_phase=coeff_correction.*field_rotate;%Correction du champ
```