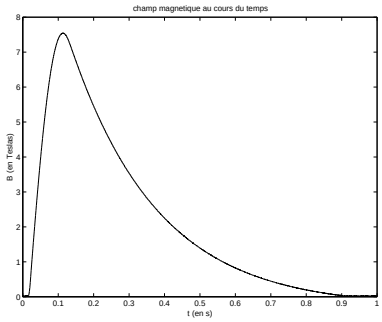
Détection Synchrone Numérique : Projet réalisé pour le L.N.C.M.P.

Samy GALLEGO
Diego MATHIEU



Mai 2002

Introduction

Dans le cadre de notre 4ème année au génie Mathématique et Modélisation de l'INSA de Toulouse, nous avons eu à effectuer un projet sur une durée de 4 mois.

Le présent document constitue un compte-rendu de ce projet, et se veut être à la fois une présentation de son cadre théorique et pratique, un manuel pour l'utilisateur du logiciel élaboré, et un guide de la programmation pour la personne désirant développer de nouvelles fonctionnalités autour de ce même logiciel.

Il s'articule en trois parties. La première est destinée à la présentation du sujet du projet et des considérations théoriques lui étant relatives. La seconde vise à commenter en profondeur les diverses options de programmation, les programmes en eux-mêmes, et donne à l'utilisateur un aperçu des interactions qui lui sont offertes avec le logiciel. Enfin la troisième a pour objectif de valider d'un point de vue pratique le programme et de donner ses caractéristiques d'exécution.

Avant d'aller plus avant dans ce compte-rendu, nous tenons à remercier toutes les personnes qui nous ont aidés pour ce projet : Bertrand Raquet et Michel Goiran, nos responsables, qui n'ont pas hésité à sacrifier beaucoup de leur temps pour nous guider et avec qui ce fût un réel plaisir de travailler ; Sylvain Jasson et Philippe Guillaume, au GMM, qui nous ont eclairci certains points théoriques obscurs et nous ont aidés à manipuler MATLAB ; et enfin tout le personnel du L.N.C.M.P. qui nous a accueilli avec beaucoup de gentillesse.

Table des matières

Première partie

Présentation générale et considérations théoriques relatives au projet

Chapitre 1

Présentation du sujet du projet

1.1 Le Laboratoire National des Champs Magnétiques Pulsés (L.N.C.M.P.) de Toulouse

Le sujet de notre projet avait été fixé par Bertrand Raquet, enseignant-chercheur attaché au L.N.C.M.P.. Cette structure existe depuis quelque 30 ans, et son but principal est la définition du comportement d'un matériau quelconque soumis à un champ magnétique très fort, ceci en vue d'établir des conclusions physiques plus large quant aux propriétés dudit matériau. Une application par exemple de ce type de recherche, est la mise en oeuvre de nouveaux types de conducteurs pour l'informatique.

Le personnel du laboratoire est composé d'une trentaine de personnes, dont une moitié est attachée au développement du dispositif expérimental (détermination du matériel utilisé, fabrication du matériel, définition de l'environnement des expériences ou réseau informatique), et l'autre moitié réalise les expériences proprement dites et les exploite sur un plan théorique. A cette seconde moitié s'ajoute une trentaine de 'visiteurs' par an, venus profiter de l'infrastructure pour mener des expériences en rapport avec leurs propres recherches. Les installations équivalentes sont relativement limitées. Il en existe seulement à Los Alamos et un laboratoire a été développé à Dresde plus récemment.

Enfin, une population de thésards, étudiants stagiaires ou menant un projet dans le cadre de leur études gravite autour des 'permanents'.

1.2 Le dispositif expérimental du L.N.C.M.P.

Il faut, avant d'entrer plus en détail dans la présentation de notre projet, nous intéresser au fonctionnement général des installations du laboratoire.

Au sous-sol du bâtiment sont installés des bancs de condensateurs pouvant stocker un équivalent énergétique de 14MJ. Ceux-ci sont reliés par l'intermédiaire d'interrupteurs à thyristors et d'un système de commutateurs

à des bobines, lesquelles sont situées au rez-de-chaussée. La décharge d'un nombre donné de condensateurs dans une de ces bobines permet de générer un champ magnétique intense, pouvant aller jusque 60 Teslas (champ magnétique de la Terre : $5 * 10^{-5}$ T), sur une durée d'environ 1 seconde.

Le terme 'pulsé' appliqué à ce champ est dû à sa forme caractéristique de pulse(cf Fig. 1).

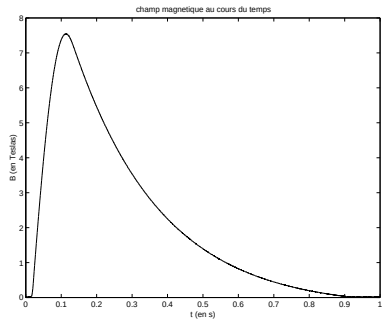


Fig. 1 : Exemple de champ magnétique obtenu au cours du temps.

L'échantillon à tester est placé au centre de la bobine, équipée d'un cryostat à ^4He. Les mesures sont en effet effectuées à basse températures, généralement de l'ordre du Kelvin, afin d'éviter les perturbations thermiques.

Cet échantillon est intégré à un circuit électrique de la forme de celui de la figure Fig. 2. On mesure la tension à ses bornes. Les perturbations dans la sinusoïde du générateur sont alors révélatrices de propriétés physiques du matériau testé.

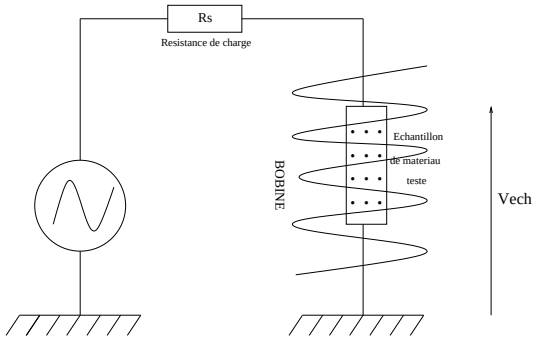


Fig. 2 : Schéma du montage électrique utilisé.

1.3 Sujet du projet

Pour pouvoir exploiter les mesures réalisées, il faut tout d'abord extraire de la tension récupérée aux bornes de l'échantillon les perturbations dues à celui-ci. Le laboratoire dispose pour cela de deux appareils effectuant une détection synchrone (cf chapitre 2). Ils fonctionnent de manière analogique,

et leurs différents composants induisent un retard, dans la transmission du signal, qu'il est difficile de quantifier, engendrant de petites erreurs en sortie.

De plus, il paraissait intéressant pour les chercheurs de tester plusieurs préfiltrages sur les mêmes données d'une experience, et de pouvoir effectuer un raffinage pour annuler des fréquences parasites par exemple, chose que la détection analogique n'autorisait pas.

Le travail proposé était alors de remplacer les appareils et leur traitement analogique des données par un traitement purement informatique et donc numérique portant sur les données acquises par un échantillonneur.

Pour mieux se rendre compte des tenants et aboutissants de cette réalisation, il nous est utile d'introduire à présent sur un plan purement théorique la détection synchrone.

Chapitre 2

Présentation théorique de la détection synchrone

2.1 Introduction

On désigne sous le nom de 'détection synchrone' un appareil de laboratoire capable d'extraire un signal continu (ou analogique) modulé en amplitude par un signal périodique lui aussi continu, de fréquence f_0 appelé 'porteuse'. On désignera tout au long de ce document par 'détection synchrone' aussi bien l'appareil physique que la fonction remplie par celui-ci.

Par ailleurs, la détection synchrone est une application découlant majoritairement de l'analyse de Fourier de fonctions. Plus encore, comme elle traite des signaux physiques, l'espace fonctionnel considéré tout au long de notre étude sera l'espace relatif à la norme énergie, soit $(L^2, ||.||_2)$ où $||f||_2 = \sqrt{\int f^2\, dt}$. Ces fonctions seront de surcroît à support compact.

L'ensemble des considérations théoriques ci-après sera fait dans l'espace plus large des distributions tempérées S'.

On s'intéressera plus particuliérement aux aspects mathématiques de la détection synchrone, en laissant de côté tous les points plus spécifiques à la physique.

2.2 La modulation d'amplitude

Soit V_i un signal continu et, par ailleurs un signal périodique x, de fréquence unique f_0. La modulation d'amplitude de x par V_i consiste simplement en la multiplication du second par le premier (connu sous le nom de 'porteuse') :

$$v(t) = V_i(t).x(t).$$

Pour illustrer notre point de vue, prenons un exemple simple :

$$\text{Soit } V_i(t) = t^2 \text{ et } x(t) = \cos(2\pi f_0\, t) \text{ avec } f_0 = 3,\ t \in [0; 2\pi].$$

La modulation de x par V_i donne alors :

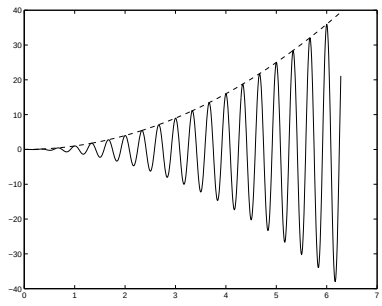


Fig. 3 : Exemple de modulation d'amplitude.

Sur la figure 3, on a tracé V_i en pointillés.

Il est nécessaire d'introduire une restriction à la définition de V_i. Dans le cadre de la détection synchrone, celle-ci se doit en effet d'avoir un spectre limité. En d'autres termes, on doit avoir $supp(\widehat{V_i}) \subset [-f_1, f_1]$ (où $\widehat{V_i}$ est la transformée de Fourier de V_i) avec $f_1 < f_0$.

Ceci nous amène à des considérations sur la modulation d'amplitude dans l'espace des fréquences. Si l'on schématise tout ce que nous avons introduit, on obtient :

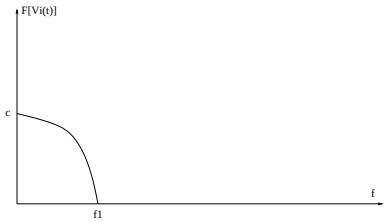


Fig. 4 : Transformée de Fourier de V_i.

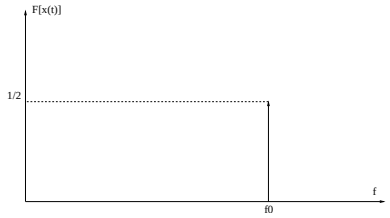


Fig. 5 : Transformée de Fourier de x.

soit pour le signal modulé en amplitude :

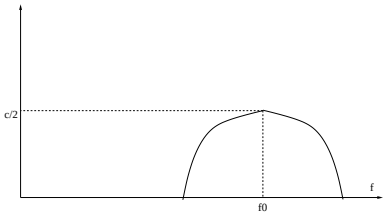


Fig. 6 : Transformée de Fourier de v.

Pour simplifier, on n'a représenté que les parties correspondant à des fréquences positives.

On peut traduire formellement ces observations. Supposons que notre porteuse soit un cosinus de fréquence f_0. La muliplication de V_i par x dans l'espace temporel se traduit par une convolution dans l'espace fréquentiel :

$$\begin{aligned}\mathcal{F}(\cos(2\pi f_0 t)V_i(t)) &= \mathcal{F}(\cos(2\pi f_0 t)) * \mathcal{F}(V_i(t)) \\ &= \tfrac{1}{2}(\delta_{-f_0} + \delta_{f_0}) * \widehat{V_i}(\lambda) \\ &= \tfrac{1}{2}(\tau_{-f_0}\,\widehat{V_i}(\lambda) + \tau_{f_0}\,\widehat{V_i}(\lambda)).\end{aligned}$$

La convolution ci-dessus est possible car $\mathcal{F}(\cos(2\pi f_0 t))$ et $\mathcal{F}(V_i(t))$ sont dans $\mathcal{E}'$, espace des distributions à support compact.

2.3 Principe de fonctionnement de la détection synchrone

Physiquement, la modulation du signal s'effectue 'naturellement', car le phénomène physique qui intéresse l'expérimentateur agit directement sur l'amplitude du signal excitateur qui n'est autre que la porteuse. On veut donc récupérer le signal V_i.

La démodulation consiste idéalement à multiplier le signal v par un signal de référence r, périodique de fréquence f_0, fréquence de la porteuse ; puis de filtrer le signal ainsi obtenu par un filtre passe-bas à fréquence de coupure f_c, avec $f_1 < f_c < 2\,(f_0 - f_1)$.

Ceci se traduit formellement dans l'espace des fréquences, si l'on reprend le signal introduit en 2.2 et en prenant $r(t) = cos(2\pi f_0 t)$, par :

$$\begin{aligned}\mathcal{F}(v(t).r(t)) &= \widehat{v}(\lambda) * \widehat{r}(\lambda) \\ &= (\tfrac{1}{2}(\delta_{-f_0} + \delta_{f_0}) * \widehat{V_i}(\lambda)) * \tfrac{1}{2}(\delta_{-f_0} + \delta_{f_0}) \\ &= \tfrac{1}{4}(\tau_{-2f_0}\,\widehat{V_i}(\lambda)) + \tfrac{1}{4}(\tau_{2f_0}\,\widehat{V_i}(\lambda)) + \tfrac{1}{2}(\widehat{V_i}(\lambda)).\end{aligned}$$

On récupérera ainsi le signal multiplié par une constante 0.5 après le filtrage passe-bas.

Remarque : On suppose ici qu'un filtre idéal, c'est à dire de fonction de transfert $H(\lambda)$ telle que :

$$H(\lambda) = \begin{cases} 1 & \text{pour } 0 \leq |f| \leq f_c \\ 0 & \text{pour } |f| > f_c \end{cases}$$

et de phase nulle partout, est réalisable.

2.4 Considérations physiques sur la détection synchrone

La détection synchrone est souvent utilisée pour traiter des signaux noyés dans du bruit. On effectue alors la modulation en amplitude par une porteuse dont la fréquence correspond à une fréquence où le bruit en transformée de Fourier est minimal. Le bruit peut avoir différentes causes, par exemple les appareils de mesures ou les interactions électromagnétiques avec l'appareillage général, comme c'est le cas au Laboratoire National des Champs Magnétiques Pulsés malgré les différentes installations mises en place pour s'en affranchir, ou l'étape d'amplification pour les signaux de faibles amplitudes.

De manière générale, le principe est utilisé dans les deux cas suivants :

- Mesure : Signal issu d'un capteur.
- Transmission : Réception d'un signal modulé.

Physiquement, l'appareil réalisant la détection synchrone possède un paramètre variable permettant de d'inclure et de modifier une phase sur le signal de référence, ceci est utile notamment dans le cas où la modulation d'amplitude s'accompagne d'un déphasage de la porteuse ainsi que nous le verrons dans le cas de notre application. Pour récupérer le signal d'origine, on fait alors appel au principe de la double détection synchrone (cf 2.5).

2.5 Mise en oeuvre de la détection synchrone dans notre projet

Comme nous l'avons déjà précisé dans le premier chapitre, le but de notre de projet était de mettre en oeuvre une détection synchrone numériquement, c'est-à-dire par un processus maniant des variables discrètes. L'information principale fournie est l'évolution au cours du temps de la tension alternative modulée en amplitude aux bornes d'un échantillon de matière soumis à un champ magnétique sur un temps d'environ une seconde. L'expérience montre que l'échantillon ne se comporte pas comme une résistance pure, mais est plutôt assimilable à un circuit de type RLC. Pour cette raison, le signal fourni est déphasé par rapport à la porteuse, dont l'échantillonnage nous est également fourni.

Pour retrouver la totalité de signal propre à l'échantillon de matière, on fait appel au principe de la double détection synchrone (cf Fig. 7).

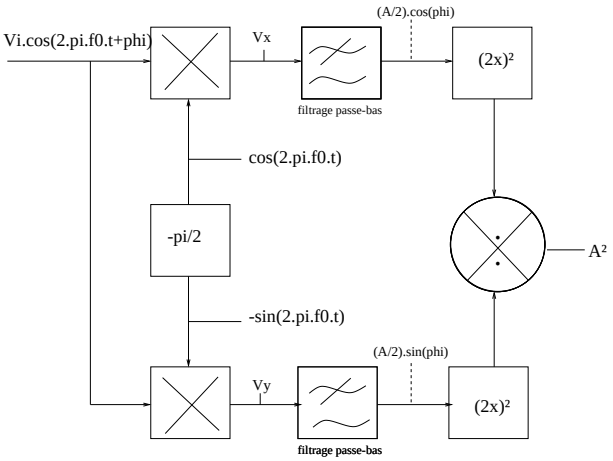


Fig. 7 : Principe de la double détection synchrone analogique.

Commentaires du schéma : On suppose que le signal en entrée est de la forme $V_i(t)\cos(2\pi f_0\, t + \phi)$, où ϕ est la phase. La porteuse est de la forme $\cos(2\pi f_0\, t)$. On considère alors deux signaux de référence, l'un consitué par la porteuse, l'autre par la porteuse retardée de $\frac{\pi}{2}$. La multiplication du signal en entrée par ces deux références donne respectivement V_x, signal en phase, et V_y, signal hors phase. On effectue le même filtrage passe-bas que dans la détection synchrone sur V_x et V_y. Le signal total caractéristique de l'échantillon testé est $S = 2 \times \sqrt{(V_x^2 + V_y^2)}$.

On peut poser ces multiplications de fonctions formellement, par exemple dans l'espace du temps :

$$V_i(t)\cos(2\pi f_0 t + \phi).\cos(2\pi f_0 t) \qquad = V_i(t).\tfrac{1}{2}(\cos(4\pi f_0 t + \phi) + \cos(\phi))$$

$$V_i(t)\cos(2\pi f_0 t + \phi).(-\sin(2\pi f_0\, t)) \quad = V_i(t).\tfrac{1}{2}((-\sin(4\pi f_0 t + \phi)) + \sin(\phi))$$

soit, après le filtrage passe-bas,

$$V_x(t) = \tfrac{1}{2}V_i(t).\cos(\phi)$$

$$V_y(t) = \tfrac{1}{2}V_i(t).\sin(\phi).$$

On peut récupérer également la phase ϕ, par exemple en posant : $\phi = \arctan(\frac{V_y}{V_x})$.

Maintenant que nous avons situé la détection synchrone sur un plan théorique, intéressons à la manière dont nous l'avons traitée dans notre projet, ainsi qu'aux facilités d'interactions utilisateur/programme mises en place.

Deuxième partie

Travail réalisé relativement à la programmation

Chapitre 3

Programmation pour le traitement de la détection synchrone sur des variables discrètes

3.1 Choix des ressources utilisées

Afin de réaliser un logiciel capable d'effectuer un traitement numérique des données fournies qui soit à la fois compatible avec les ressources informatiques du L.N.C.M.P., capable d'effectuer un traitement mathématique consistant pour la plupart en des manipulations de vecteurs, mais qui soit également portable, pas trop formalisé et facile d'accès, nous avons décidé d'effectuer l'ensemble de la programmation en MATLAB. Ce logiciel est en effet destiné aux opérations portant sur des matrices, est suffisamment développé pour permettre des échanges avec l'utilisateur (du type entrées de variables), et surtout possède l'extension TOOLBOX 'SIGNAL ANALYSIS' dont les nombreuses fonctionnalités sont largement en rapport avec le sujet qui nous intéresse.

3.2 Considérations générales relatives à la programmation

3.2.1 Données en entrée

Il est important de rappeler dans cette section le type de données auxquelles on a affaire pour justifier les choix effectués dans la programmation.

Ainsi, notre programme devait traiter un fichier d'entrée contenant sous la forme de vecteurs colonnes : le champ magnétique, la porteuse et le signal

modulé, chacun échantillonné sur une durée fournie (cf 5.2.2) d'environ une seconde, à une fréquence variable allant de 10 KHz à 100 KHz, soit un vecteur de 10000 à 100000 points. La fréquence de la porteuse varie de 200 Hz à 5000 Hz.

La mise en oeuvre de la détection synchrone numérique demande tout d'abord d'adapter les différentes étapes du processus au traitement de vecteurs. Intéressons nous tout d'abord au cas des filtres.

3.2.2 Les filtres

Rappels théoriques sur les filtres discrets :

Un signal discret est par définition des distributions tempérées de la forme :

$$x = \sum_{n \in \mathcal{Z}} x_n \delta_n, \quad x_n \in \mathcal{C},$$

et qui résulte généralement de l'échantillonnage d'un signal à temps continu. Ici, nous restreignons la définition en supposant $x_n \in \mathcal{R}$ et $n \in [1, .., N]$ (on travaille à temps fini), et nous remplaçons la notation formelle de x par sa notation sous forme de vecteur : $x = (x_0, ...x_N)^T$.

Remarque : On commmence à numéroter les indices de x à partir de 0 dans tout ce rappel théorique. Il faudra éviter les confusions avec les notations MATLAB utilisée par la suite.

Les filtres discrets sont des filtres de la forme $x \mapsto y = h * x$ où x est un signal dicret et h une distribution tempérée de la forme :

$$h = \sum_{n \in \mathcal{Z}} h_n \delta_n, \quad h_n \in \mathcal{C}.$$

Les filtres discrets sont répartis en deux grandes catégories. La première est constituée des filtres à réponse impulsionnelle finie (filtres RIF), pour lesquels le dénominateur de la transformée en z $H(z)$ est de degré 0, et par conséquent $n \in [0, .., r]$. La seconde répertorie les filtres à réponse impulsionnelle infinie (filtres RII), pour lesquels le dénominateur de la transformée en z a un degré supérieur ou égal à 1, et le nombre de h_n non nuls est infini. Le concept de transformée en z étant au-delà du sujet de ce document, le lecteur intéressé pourra se référer à [2].

Enfin, on constate que le filtrage discret consiste en une convolution de deux vecteurs. Ainsi, si x a une longueur n et h une longueur p, alors y a comme longueur $n + p$.

Prise en compte des caractéristiques du filtrage dans la programmation :

Dans notre programme, on est amené à utiliser de nombreux filtrages, non seulement pour la détection synchrone, mais aussi pour extraire le maximum d'informations sur le signal démodulé et limiter les perturbations dues aux bruits (cf 3.2.4). Par ailleurs, comme on manipule différents vecteurs, amenés à subir des opérations différentes, on veut conserver des tailles identiques à leurs tailles initiales.

On prend alors en compte le retard de groupe du filtre. Ce phénomène est analogue au cas continu.La solution envisagée est de calculer ce dernier noté r_g lors de l'établissement des coefficients du filtre (cf 5.2.3), de donner au vecteur résultat une taille $n + r_g$, où n est la taille du vecteur à filtrer, et enfin de supprimer les r_g premières valeurs du vecteur résultat après le filtrage.

Un autre phénomène nuisible est celui de Gibbs. Prenons l'exemple d'un filtre passe-bas, de transformée de Fourier ci-après (Fig. 8) :

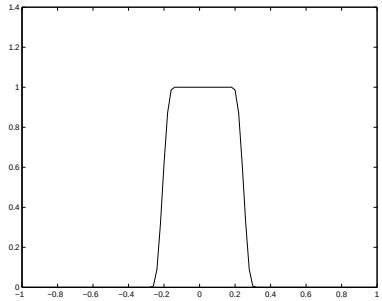


Fig. 8 : module de la TF d'un filtre passe-bas.

Lorsque la pente de ce filtre devient trop abrupte, on peut assister à la formation d'oscillations au voisinage de la fréquence de coupure sur les valeurs de la transformée de Fourier du vecteur résultat. Ces oscillations apparaissent lors de la convolution de x et h. C'est pourquoi on veillera lors de l'utilisation des filtres présentés dans la suite du document à ne pas prendre des fréquences de coupures aux voisinages de fréquences porteuses d'une information qui nous intéresse.

Choix des filtres :

Connaissant maintenant les principales caractéristiques d'un filtrage discret, et étant convaincus que la programmation nécessitera l'emploi d'un grand nombre de filtres, nous avons maintenant à choisir parmi la pléthore de filtres que nous offre la TOOLBOX 'SIGNAL ANALYSIS' de MATLAB.

Les exigences sont assez souples, le temps de calcul ne devant pas être nécessairement minimisé, et les fréquences de travail, ainsi que nous le verrons plus tard, ne sont pas très resserrées.

La première question qui se pose est : filtre RIF, ou filtre RII ? Les essais réalisés ont montré que le temps de calcul nécessaire au filtrage d'un vecteur explose pour les filtres RII tels que butterworth(cf [1]). %reference bibliographique au guide de la toolbox signal analysis. Ceci est largement dû au fait que nos expérimentations se déroulent à des degrés élevés, de par les caractéristiques des siganux échantillonnés (en particulier : fréquence d'échantillonnage et rapport de la fréquence d'échantillonnage sur la fréquence de la porteuse, comme expliqué en 5.2.3).

Il est évident que les filtres RII s'approchent plus des filtres idéaux que les filtres RIF, mais le résultat obtenu avec ces derniers est largement acceptable ici.

Nous avons ensuite focalisé notre choix sur les filtres que nous avions déjà employé dans des T.P. antérieurs : fir1 et fir2. Nous avons opté pour le deuxième car il permet de fixer des valeurs de la transformée de Fourier du filtre à des fréquences relatives données. Il offre ainsi la possibilité d'un plus grand contrôle sur le filtre que fir1 qui a tendance à trop lisser les contraintes données.

Exemple d'utilisation : la suite d'instructions :

```
freq=[0 0.2 0.3 1]; %on fixe les frequences pour lesquelles
                    % on veut maitriser les valeurs de la TF du filtre
val=[1 1 0 0]; %on fixe les valeurs aux frequences donnees ci-dessus
deg=100; %on fixe le degre du filtre
b=fir2(100,freq,val); %on forme les coefficients du filtre
plot(abs(fft(b))); %on visualise la TF du filtre forme
```

donnera la figure suivante (Fig. 9) :

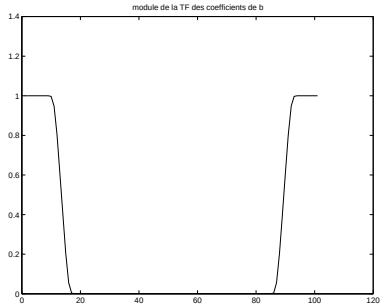


Fig. 9 : module de la TF du filtre.

On constate que le degre du filtre est en fait le nombre de points utilisés pour le décrire. Par ailleurs, lorsqu'on trace le module de sa transformée de Fourier, celui-ci est symétrique par rapport à sa valeur milieu. Ceci est dû au fait que ce qui constitue la seconde partie du filtre correspond aux fréquences négatives (si l'on considère toujours la TF).

Il peut être alors utile de donner le module de la transformée tracé en fonction des fréquences relatives (cf Fig. 10).

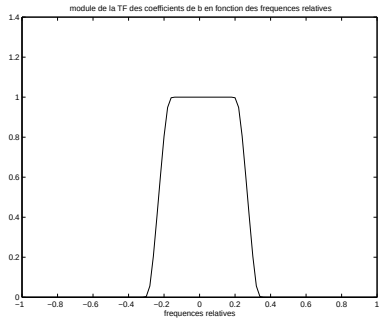


Fig. 10 : module de la TF en fonction des fréquences relatives.

Enfin, la dernière considération utile à cet exposé a trait au retard de groupe du filtre introduit plus haut. fir2 renverra des coefficients définissant l'équivalent discret d'un filtre à phase linéaire. Même si en pratique, cela ne se révèle pas exactement vrai, on continuera à faire une approximation linéaire, ce qui nous donnera un retard de groupe unique pour chaque filtre, que l'on détermine comme expliqué dans 5.2.3.

La fonction MATLAB freqz nous permet de tracer la phase en degré pour le filtre défini ci-dessus, ainsi que son module de sa TF en dB, comme le montre la figure Fig. 11.

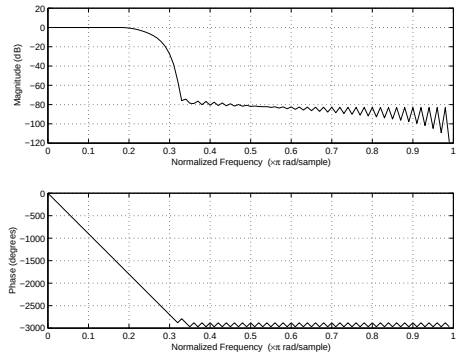


Fig. 11 : Module en dB et phase du filtre défini ci-dessus (freǵuences positives).

3.2.3 Multiplication par les deux références de la double détection synchrone

Les relevés de mesures fournis ont été faits à partir d'une porteuse considérée comme un cosinus parfait. La multiplication du signal modulée par deux références présentée avec la double détection synchrone consiste à effectuer le produit du vecteur signal modulé par un cosinus puis par un sinus de même fréquence.

La première solution envisagée a été de construire nous même, de manière exacte, ces fonctions après avoir établi expérimentalement la fréquence précise de la porteuse à partir de la fréquence donnée. Toutefois, les fréquences ainsi déterminées au centième d'Hertz près n'avaient pas encore une précision suffisante. En effet, le nombre d'oscillations sur une seconde de la porteuse variant entre 200 et 5000, le décalage entre le cosinus théorique construit et la porteuse était toujours supérieur à une période pour $t = 1$.

La composante continue de la porteuse :

Nous nous sommes alors décidés à nous servir directement de la porteuse comme cosinus multiplicateur. Un autre constat a alors été mis en évidence : La porteuse comporte une composante continue et son amplitude est différente de 1. Il nous faut alors reconsidérer la théorie de la détection synchrone.

Une composante continue peut être considérée comme un cosinus de fréquence nulle. Sa transformée de Fourier est alors $\delta \times c$, où c est la valeur de la composante continue. On peut alors réécrire la modulation en amplitude de la porteuse dans l'espace des fréquences :

$$\begin{aligned}
\mathcal{F}((a\,\cos(2\pi f_0 t) + c)V_i(t)) &= \mathcal{F}(a\,\cos(2\pi f_0 t) + c) * \mathcal{F}(V_i(t)) \\
&= \tfrac{1}{2}(a\,\delta_{-f_0} + a\,\delta_{f_0} + 2c\,\delta) * \widehat{V_i}(\lambda) \\
&= \tfrac{a}{2}(\tau_{-f_0}\,\hat{V_i}(\lambda) + \tau_{f_0}\,\widehat{V_i}(\lambda)) + c\,\widehat{V_i}(\lambda),
\end{aligned}$$

où a est l'amplitude de la porteuse.

Il nous faut donc nous affranchir de cette composante indésirable en $\lambda = 0$. Pour cela, avant de faire la multiplication du signal modulé par la porteuse, on cherche la valeur de la composante continue de la porteuse, et on recentre celle-ci.

Remarque : Dans la pratique, on cherche également l'amplitude a de la porteuse, car celle-ci n'est pas égale à 1. Lors du recentrage, on soustrait au vecteur de porteuse un vecteur de même taille et de valeur unique égale à c. On divise chacune des composantes du vecteur ainsi obtenu par l'amplitude de la porteuse. On obtient ainsi un cosinus c_c oscillant entre 1 et -1.

La multiplication par cette référence du signal modulé donne dans l'espace des fréquences :

$$(\tfrac{a}{2}(\tau_{-f_0}\,\widehat{V_i}(\lambda)+\tau_{f_0}\,\widehat{V_i}(\lambda))+c\,\widehat{V_i}(\lambda))*(\mathcal{F}(\cos(2\pi\,f_0 t)))$$

$$(\tfrac{a}{2}(\tau_{-f_0}\,\widehat{V_i}(\lambda)+\tau_{f_0}\,\widehat{V_i}(\lambda))+c\,\widehat{V_i}(\lambda))*(\tfrac{1}{2}(\delta_{-f_0}+\delta_{f_0}))$$

$$\tfrac{a}{4}(\tau_{-2f_0}\,\widehat{V_i}(\lambda)+\tau_{2f_0}\,\widehat{V_i}(\lambda))+\tfrac{c}{2}(\tau_{-f_0}\,\widehat{V_i}(\lambda)+\tau_{f_0}\,\widehat{V_i}(\lambda))+\tfrac{a}{2}(\widehat{V_i}(\lambda)).$$

On s'aperçoit que le signal centré en 0 est conforme à nos attentes, mais aussi qu'un signal indésirable apparaît en f_0. Il faudra donc envisager un filtrage passe-bas en sortie plus serré que prévu.

Etablissement de la seconde référence :

On a donc trouvé la référence qui nous permettra d'extraire V_x, le signal en phase. Pour la référence relative au signal hors-phase, il suffit alors d'induire un retard de $\frac{\pi}{2}$. Pour ce faire, on cherche le nombre de points n_p correpondant à un décalage d'un quart de période à partir de la fréquence de la porteuse donnée en entrée (cf 5.2.2). On tronque les n_p premiers points du cosinus de référence c_c (que nous venons d'établir) et du signal modulé, et on construit le sinus de référence en supprimant les n_p derniers points de c_c. On conserve donc la compatibilité des longueurs entre les vecteurs.

La multiplication elle-même entre le signal modulé et les deux références se fait simplement en multipliant composante à composante le vecteur signal modulé tronqué et cosinus de référence, puis le vecteur signal modulé tronqué par l'opposé du sinus de référence.

3.2.4 Atténuation du bruit

Comme on l'a déjà souligné, le signal modulé que nous devons traité est noyé dans le bruit. Ce bruit est réputé être gaussien, ce qui se traduit dans l'espace des fréquences par une répartition aléatoire et de limite uniforme lorsque le nombre de points échantillonnés tend vers l'infini..

Pour retrouver dans le signal démodulé une image aussi proche que possible de ce qu'il fût dans la réalité, on a alors intérêt à effectuer une série de filtrages rapprochés autour des fréquences supposées de ce signal pour s'affranchir du bruit. Toute la difficulté consiste à différencier ce qui est bruit de ce qui est signal, et ceci est laissé au libre arbitre de l'utilisateur.

Chapitre 4

L'interface du logiciel

4.1 Le cahier des charges

Comme toute bonne interface graphique, nous avions le devoir de faire une interface facile d'utilisation, adaptée aux besoins de l'utilisateur et dotée d'un nombre suffisant de paramètres. Nous n'avions pas un véritable "cahier des charges" mais les chercheurs du L.N.C.M.P. nous ont plutôt guidés au fur et à mesure, en nous demandant des fonctionnalités qu'ils aimeraient voir apparaître sur l'interface. Cette manière de travailler nous a obligés à faire un programme relativement modulable.

4.2 Les possibilités de MATLAB6

MATLAB6 prévoit un ensemble de commandes et fonctions dont l'utilisation est relativement aisée pour la création et la manipulation d'objets graphiques (fenêtres, menus, boutons de commande, cases à cocher, etc. . .).

L'élément de base d'une interface graphique est la fenêtre. Elle permet de grouper des outils graphiques dans un même cadre dans un but de clarté et de manipulation facile. A une fenêtre sont associées des propriétés telle que la taille, le nom, la position, le titre, la couleur, etc... Chaque objet graphique possède un identificateur unique (appelé pointeur, " handle " ou poignée) qui lui est attribué lors de sa création. On peut consulter les valeurs courantes des propriétés d'un objet par la commande **get**. Typiquement, on aura la ligne de commande :

```
V=get(hobj,'Nompropriete');
```

Il est par ailleurs possible de modifier les valeurs de certaines propriétés à l'aide de la commande **set** :

```
V=set(hobj,'Nompropriete','Valeurpropriete');
```

Les contrôles sont des objets graphiques qui réagissent et provoquent une action lorsqu'ils sont manipulés par la souris ou le clavier. Nous avons utilisé les contrôles suivants :

Bouton poussoir (*PushButton*) :

Case à cocher (*CheckBox*) :

Textes à modifier (*Edit*) :

Pour créer des contrôles, on utilise la fonction **uicontrol** en respectant la syntaxe comme dans cet exemple :

```
FFTVI = uicontrol('Units','points', ...
'BackgroundColor',[0 0 0], ...
'foregroundColor',[1 1 1], ...
'Position',[15 90 150 25], ...
'String','graphe de la FFT du signal module', ...
'Style','checkbox');
```

Les propriétés *'BackgroundColor'* et *'ForegroundColor'* sont suivies de vecteurs contenant les valeurs des trois couleurs fondamentales *'[R G B]'*.

La propriété *'Position'* est suivie du vecteur contenant la position dans la fenêtre et la taille du contrôle *'[X Y tailleX tailleY]'*.

'String' contient la chaîne de caractère qui sera écrite.

'Style' indique le type de contrôle.

4.3 Le choix de présentation de l'interface

Nous avons décidé de partager l'interface en quatre sections différentes :

1. Les informations relatives au **fichier** données en entrée :
   - le nom du fichier en entrée (*data*)
   - le numéro de colonne de la porteuse (*colport*)
   - le numéro de colonne du signal (*colsig*)
   - le numéro de colonne du champ magnétique (*colmag*).
2. Les informations relatives au **signal** à démoduler :
   - la fréquence de la porteuse (*f0*)

- la fréquence d'échantillonnage (*fs*)
- la durée de la mesure (*t*).

3. Les informations relatives au **traitement** de la détection synchrone :
   - Pour le préfiltrage du signal (tous les filtres sont en option) :
     - la fréquence de coupure du filtre passe-bas (*fc1*)
     - les deux fréquences de coupure du filtre passe-bande (*fc11* et *fc12*)
     - les deux fréquences de coupure du filtre coupe-bande (*fc13* et *fc14*).
   - la fréquence de coupure utilisée lors de la détection (*fc*)
   - l'option de tracé des graphes de la transformée de Fourier du signal modulé et du signal démodulé
   - l'option de tracé des graphes détaillés (cf 4.5.3)
   - l'option de lissage par interpolation linéaire du champ magnétique ainsi que son facteur de réduction (*facteur1*)
   - le nom du fichier en sortie (*datasortie*).
4. les informations relatives au **raffinage** des résultats :
   - Pour le postfiltrage du signal démodulé :
     - la fréquence de coupure du filtre passe-bas (*fc2*)
     - les deux fréquences de coupure du filtre passe-bande (*fc21* et *fc22*)
     - les deux fréquences de coupure du filtre coupe-bande (*fc23* et *fc24*)
     - la fréquence de coupure du filtre passe haut (*fc3*).
   - le facteur de réduction du nombre de points (*facteur2*)
   - l'option de tracé du graphe de la transformée de Fourier du signal démodulé et postfiltré
   - le nom du fichier en sortie (*datasortie2*).

On peut voir ci-dessous l'interface graphique que l'on obtient en lançant le programme detection.m :

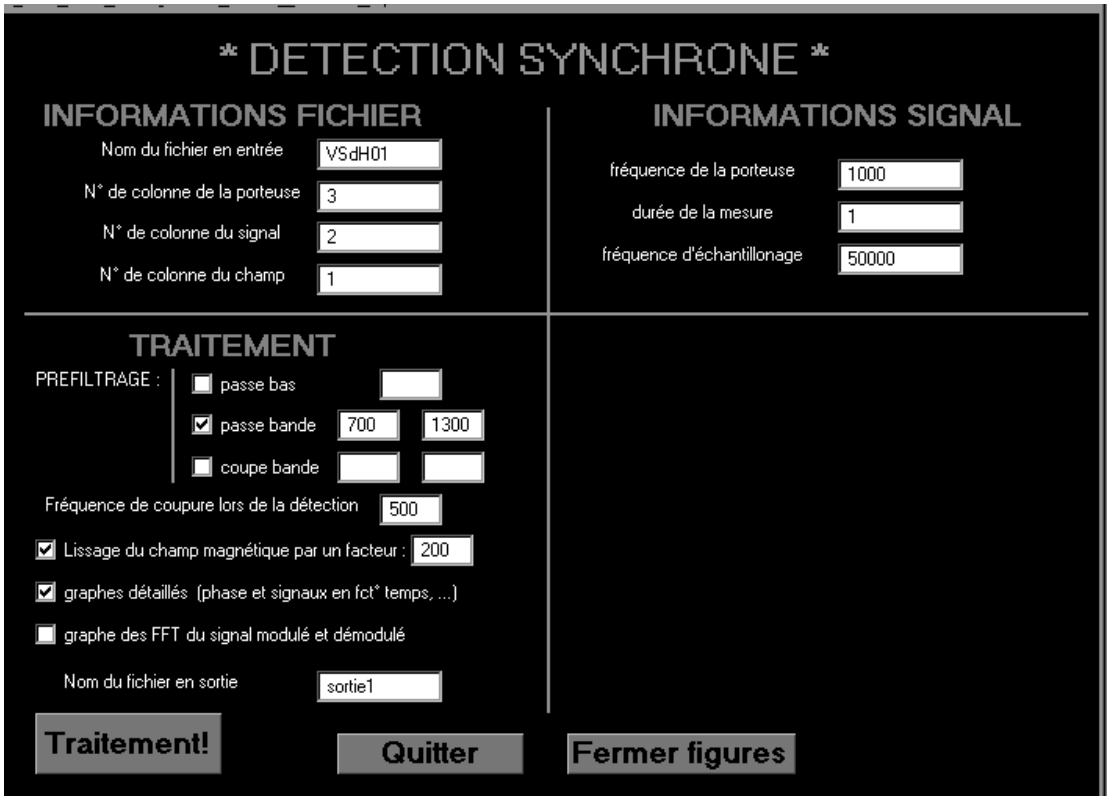


Fig. 12 : Interface apparaissant lors du lancement de detection.m.

Lors du lancement, toutes les options du raffinage sont cachées puisque l'on n'a pas de données traitées à raffiner. Pour cela, on utilise la commande :

```
set(HANDLE,'visible','off');
```

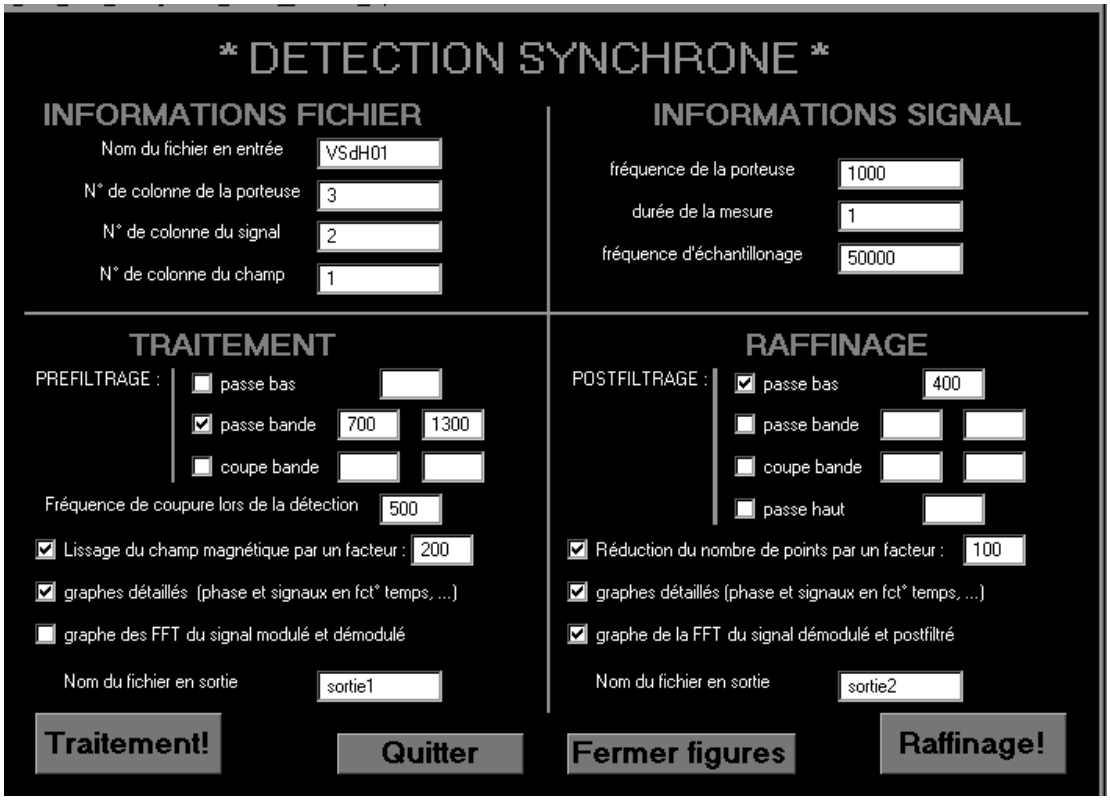


Fig. 13 : Interface apparaissant après le premier traitement.

Après le premier traitement, ces options apparaissent automatiquement en utilisant la commande :

```
set(HANDLE,'visible','on');
```

4.4 Fonctionnement général du programme

Voici comment s'organisent les différents fichiers matlab que nous avons créés :

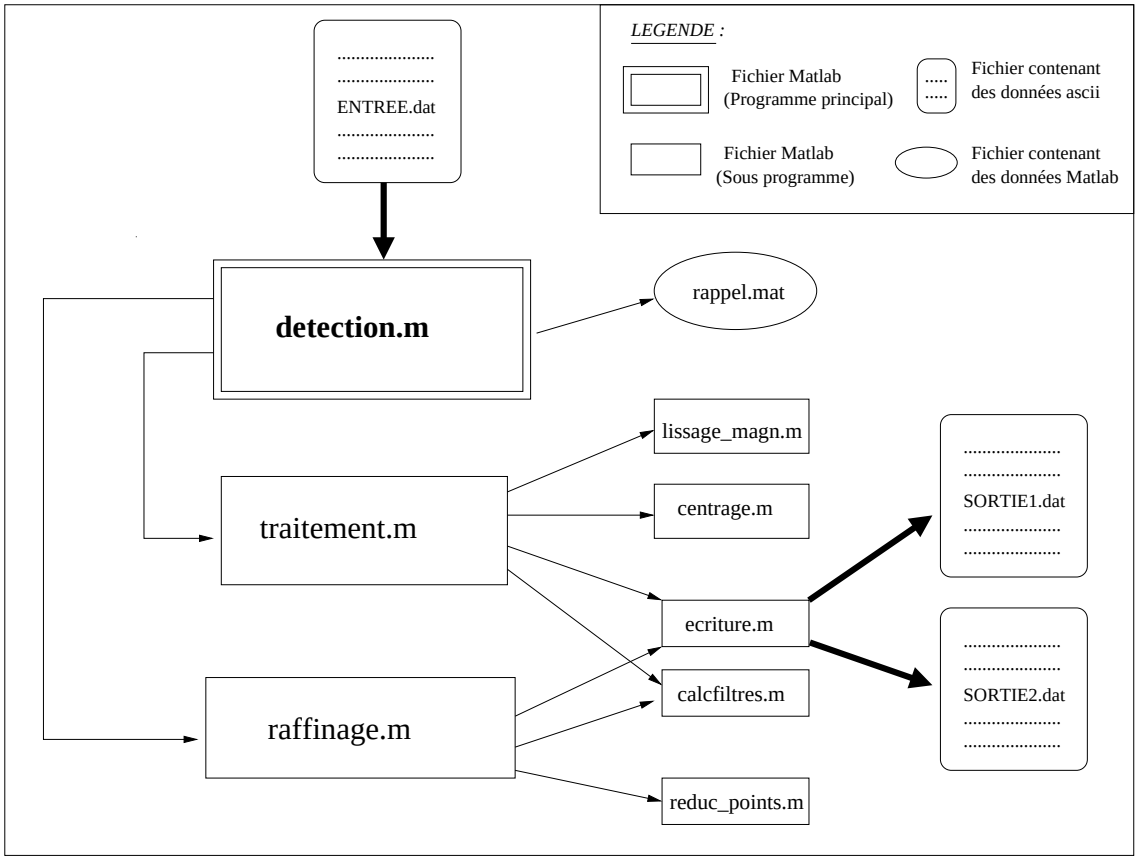


Fig. 14 : Fonctionnement du logiciel.

Nous verrons plus loin le détail du code contenu dans ces fichiers (cf chapitre 5).

4.5 Les options supplémentaires de l'interface graphique

4.5.1 Protection des fichiers en écriture

Pour éviter que l'utilisateur écrase un fichier contenant des données importantes, nous avons utilisé une protection grâce au fichier ecriture.m. Ce fichier permet de verifier de façon récursive si le nom de fichier rentré par l'utilisateur existe déja. Pour cela, nous avons utilisé la fonction **exist** de MATLAB qui renvoie 1 si le fichier existe et 0 sinon. Tant que l'utilisateur

tape un nom de fichier existant et jusqu'à ce qu'il clique sur le bouton de commande "ignorer", la fenêtre suivante s'ouvre :

Fig. 15 : Fenêtre destinée à protéger les fichiers en écriture.

4.5.2 Affichage des données rentrées par le dernier utilisateur

Pour ne pas avoir à retaper toutes les informations nécessaires à l'utilisation du programme à chaque nouveau lancement de detection.m, nous enregistrons à chaque lancement detraitement.m et raffinage.m une sauvegarde des variables utiles (avec la commande **save** rappel.mat) dans un fichier rappel.mat. Et à chaque fois que l'on relance l'interface graphique, ces données sont rechargées et repositionnées dans les cases adéquates (avec la commande **load** rappel.mat puis **set** comme l'explique le paragraphe 4.2).

4.5.3 Superposition des graphes

Une option utile à l'utilisateur permet de superposer les graphes obtenus à chaque lancement detraitement.m et raffinage.m.

Si l'option "détail des graphes" n'est pas cochée, une seule fenêtre graphique s'ouvre et elle fait apparaître les graphes des signaux en phase et hors phase ainsi que le signal total en fonction du champ magnétique. C'est finalement le résultat le plus intéressant à étudier d'après ce que nous ont dit les chercheurs du L.N.C.M.P..

En revanche, si l'option "détail des graphes" est cochée, l'utilisateur aura en plus à sa disposition les graphes suivants (dans l'ordre d'apparition) :

1. sur une même figure, le signal modulé et démodulé en fonction du temps (très pratique pour voir si le signal démodulé colle bien à l'enveloppe du signal modulé cf 6.2)
2. le signal en phase en fonction du temps
3. le signal hors phase en fonction du temps
4. le signal total en fonction du temps
5. la phase en fonction du temps
6. le signal total en fonction du champ magnétique avec deux couleurs différentes pour les parties "montée" et "descente" du champ.

Un tableau de caractères contenant les couleurs pour chaque numéro d'utilisation permet de repérer facilement les différences obtenues avec les

différents traitements et raffinages (on pourra voir des exemples de graphes dans le chapitre 7).

4.5.4 Fermeture de tous les graphes

Nous avons inclu un bouton de contrôle qui permet de fermer tous les graphiques, si l'utilisateur considère que les graphes sont trop chargés et qu'il veut recommencer à zéro.

Chapitre 5

Commentaires des fonctions du programme

Tout au long de ce chapitre, nous nous appliquerons à commenter l'ensemble du code, fonction par fonction , afin que l'utilisateur ait la possibilité, s'il le veut, d'ajouter des fonctionnalités au logiciel. Les fonctions seront présentées dans l'ordre chronologique de leur apparition dans le code du logiciel, et suivant leur lieu d'appel. Pour chacune, on donne d'abord la forme d'appel, on explicite son rôle, on commente ensuite les variables en entrée et en sortie et, au besoin, on complète par quelques considérations destinées à justifier les choix effectués.

5.1 La fonction detection.m

Appel :

```
detection;
```

Rôle :
Elle permet de générer l'interface présentée au chapitre précédent et de gérer les appels à traitement et raffinage.

Commentaires : C'est la fonction centrale du logiciel. L'interface permet à l'utilisateur de rentrer les variables utiles au logiciel. Les variables nécessaires aux sous-fonctions sont déclarées GLOBAL et fixées par l'utilisateur, mais ne seront évaluées qu'à l'intérieur des fonctions.

5.2 La fonction traitement.m

Appel :

```
traitement;
```

Rôle :

Elle contrôle l'ensemble des étapes de la détection synchrone.

Commentaires :

Elle est activée par l'utilisateur grâce au bouton **'Traitement'** de l'interface.

Comme on vient de le préciser, les variables en entrée sont passées implicitement.

La première chose faite par cette fonction est d'afficher dans l'interface graphique la partie relative au raffinage (cf 5.3), qui la suit logiquement. Ensuite elle évalue les différentes variables entrées par l'utilisateur. Enfin, elle charge le fichier passé en entrée. On rappelle que celui-ci contient les vecteurs champ magnétique, la porteuse et le signal aux bornes de l'échantillon, respectivement stockés dans les variables *magn*, *r* et *Vi*. On génère un vecteur x qui prend les valeurs du temps, de 0 au temps de fin d'expérience entré par l'utilisateur.

L'acquisition des données faites, on peut passer à des manipulations numériques.

On appelle lissage_magn, si l'utilisateur a sélectionné l'option.

5.2.1 La fonction lissage_magn.m

Appel :

```
champ_lisse=lissage_magn(champ,p);
```

Rôle :

Elle lisse le champ magnétique en entrée afin d'éliminer les perturbations dues au bruit.

Commentaires :

Comme on cherche à supprimer un bruit, nous avions en premier lieu opté pour un filtrage passe-bas du champ magnétique. Le résultat avait alors présenté des oscillations indésirables, puisque le champ magnétique lissé est destiné à servir d'abscisse dans un graphe. C'est pourquoi on voudrait qu'il soit bijectif par rapport au temps, respectivement pour la montée et la descente (cf Fig 1, 1.2).

Nous avons finalement choisi d'effectuer une approximation $\mathcal{P}1 \cap \mathcal{C}^0$ en moyennant *champ* par groupes de p points. On extrapole ensuite les points obtenus par des segments de droite pour récupérer en sortie un vecteur *champ_lisse* de même taille que *champ*. On souligne le fait que chacune des moyenne de p points se trouvera à l'indice médian de ces p points dans *champ_lisse*. Le premier et le dernier point de *champ_lisse* sont le premier et le dernier point de *champ*.

Ce principe de programmation ne permet pas toujours d'obtenir la bijection souhaitée, notamment dans les parties constantes de début et fin de champ si le bruit inclut une composante basse fréquence. Le remède consiste

alors à augmenter le paramètre p, si cela ne nuit pas trop à la qualité de l'approximation.

La fonction traitement fait ensuite appel à la fonction centrage.

5.2.2 La fonction centrage.m

Appel :

```
[decalage,comp_cont,amplitude,p_centre]=centrage(porteuse,freq0,freq_ech,tps_ech);
```

Rôle :

Elle recentre la porteuse et lui redonne une amplitude égale à 1, comme dans 3.2.3.

Commentaires :

Dans 3.2.3, on suppose que la porteuse est un cosinus parfait. Toutefois, une erreur d'acquisition du signal physique peut survenir. Dans ce cas, on transforme *porteuse* en cosinus en tronquant le signal afin de poser la première valeur maximale comme correspondant à $p\_centre(1)$. Pour conserver la compatibilité des longueurs des vecteurs Vi et x avec la nouvelle porteuse dans traitement, on s'inspirera de la longueur de p_centre, avant de tronquer le début des vecteurs.

On établit l'amplitude et la valeur composante continue, respectivement *amplitude* et $comp\_cont$, en effectuant trente recherches successives, espacées entre elle de dix périodes, des maxima et minima de *porteuse* et en les traitant. On détermine le nombre de points d'une période en divisant f_ech, fréquence d'échantillonnage, par *f0*, fréquence de la porteuse. tps_ech est la durée d'échantillonnage.

On obtient alors p_centre en soustrayant $comp\_cont$ à chacune des valeurs de *porteuse* et en les divisant ensuite par *amplitude*. La variable *deca* doit fixer le nombre de points à supprimer au début ou à la fin de *porteuse*, pour établir respectivement la première et la seconde référence comme expliqué dans 3.2.3. On pose alors *deca* égal au nombre de points d'une période divisé par 4. L'ennui d'une telle méthode, surtout si *f0* est important par rapport à f_ech, c'est que l'on peut avoir un décalage approximatif, et donc un recouvrement des signaux hors-phase et en phase, dû à la perte de leur orthogonalité.

Les deux troncatures nécessaires à la compatibilité des longueurs sont faites dans traitement, tout comme l'établissement des deux références. Cette troncature ne détruit pas d'information importante, puisqu'elle est réalisée sur les premiers points du signal, lorque le champ est encore nul, soit avant le début de l'expérience proprement dite.

L'étape suivante de traitement consiste à effectuer un préfiltrage du signal modulé. Avant toute chose on calcule les coefficients du filtre employé.

5.2.3 La fonction calcfiltres.m

Appel :

```
[retardgroupe,b]=calcfiltres(nentr,fcar,fech,style);
```

Rôle :

Elle calcule avec la fonction fir2 de MATLAB les coefficients du numérateur de la transformée en z du filtre voulu.

Commentaires :

Cette fonction peut aussi bien 'fabriquer' un filtre passe-bas, passe-haut, passe-bande ou coupe-bande. Le genre du filtre est passé par une chaîne de caractères, la variable d'entrée *style*. On peut également présenter les autres variables d'entrée :

- *fech* : la fréquence d'échantillonnage des valeurs lors de l'acquisition physique des données.
- *nentr* : c'est un paramètre entier fixant le degré minimal du filtre.
- *fcar* : c'est un vecteur de taille variable fixant les fréquences caractéristiques absolues du filtre désiré. Dans le cas d'un filtre passe-bas ou passe-haut, il contient simplement la fréquence de coupure, dans le cas d'un filtre concernant une bande de fréquence, il contient la fréquence centrale et la demi-largeur de bande du filtre (cf Fig 16).

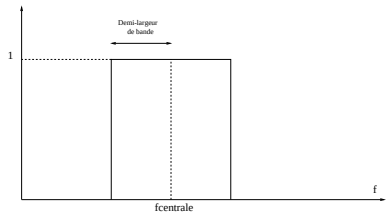


Fig. 16 : TF théorique d'un filtre passe-bande

En premier lieu, on divise la fréquence d'échantillonnage par 2 afin d'établir les fréquences relatives propres au filtre, pour tenir compte de la symétrie de ce dernier. Puis, selon la valeur prise par *style*, on crée les vecteurs contenant d'une part les fréquences relatives caractéristiques et d'autre part les valeurs leurs correspondant(cf utilisation de fir2, dans 3.2.2).

Remarque importante : On peut s'attarder ici sur nos choix quant au calcul du degré des filtres. Le filtrage discret consistant en une convolution, il est évident que nous aurons intérêt à le prendre minimum, pour limiter le nombre d'opérations. Toutefois, MATLAB réalise également un fenêtrage (type Hamming par défaut pour fir2) lors d'un filtrage, accompagné d'une périodisation. Ceci nous oblige déjà à choisir un degré au moins égal au nombre de points décrivant une période du signal, soit le rapport de la fréquence d'échantillonnage sur la fréquence de la porteuse. On fixe également un degré minimal de 100 pour tous les filtres. Ces deux considérations permettent d'établir *nentr*. Par ailleurs, pour avoir des filtres suffisamment

bien décrits, afin d'éviter les phénomènes d'atténuations et donc de pertes d'informations aux voisinages des fréquences de coupure, on cherche à ce que la plus petite partie du filtre, soit le plus petit écart entre deux fréquences relatives entrées à fir2, contienne au moins 2 points pour décrire la variation associées.

La fonction calcfiltres réalise donc le calcul de ce degré, avant de déterminer les coefficients caractéristiques du filtre, stockés dans b. Le retard de groupe d'un filtre discret étant égal au nombre de points constituant le filtre diminué de 1, la symétrie de celui-ci donne :

```
retardgroupe=floor(degre/2);
```

Suite de la fonction traitement :

Les coefficients du filtre sont connus, on peut procéder au préfiltrage de Vi dans traitement. Celui-ci se fait simplement, après avoir crée le filtre voulu par l'utilisateur et transmis au moyen de l'interface, avec la fonction filter de MATLAB, en tenant compte du retard de groupe (cf 3.2.2). Le but premier de ce préfiltrage est d'éliminer les perturbations dues à la présence d'éventuelles harmoniques de la porteuse, c'est-à-dire des cosinus à des fréquences multiples entiers de f_0 s'ajoutant au cosinus initial. Il est évident que les références étant construites à partir de la porteuse elle-même, la multiplication du signal par les références, dans l'espace des fréquences, recentrerait le signal cherché multiplié par une constante indéterminable.

L'étape suivante faite dans traitement est la multiplication de Vi par les deux références construites plus haut. Pour éliminer la constante 0.5 intervenant *a priori* (cf 2.5), on multiplie directement par le double des références.

Pour obtenir les signaux en phase et hors phase définitifs, on effectue un filtrage passe-bas, à la fréquence entrée par l'utilisateur, par le même processus que celui présenté dans le préfiltrage. On effectue alors le calcul de la phase existant entre le signal modulé et la porteuse grâce à la formule présentée dans 2.5, et on établit le signal démodulé total.

Remarque 1 : A la demande de nos responsables, nous n'avons pas cherché à diviser le vecteur signal total obtenu par l'amplitude initiale de la porteuse. Ceci est justifié par le fait que le signal démodulé est une tension, et que l'intérêt des chercheurs est plus dans la variation de la resistivité du matériau, donnée par $U = R.I$, mais que l'on ne peut obtenir directement.

Remarque 2 : L'ensemble des considérations porté dans 5.2.2 quant à l'incertitude sur l'exactitude des deux références doit amener l' utilisateur à être critique vis à vis de la phase absolue. Toutefois, des observations sur les variations relatives de la phase se placent au delà de la présente réserve.

Remarque 3 : La phase étant établie par l'intermédiaire de la fonction arctangente, elle sera comprise entre $-\pi/2$ et $\pi/2$. Elle est par ailleurs définie modulo π.

La suite de traitement nous permet de tracer les graphes ou d'effectuer les transformées de Fourier à la demande de l'utilisateur.

Enfin, on appelle la fonction ecriture pour créer le fichier de données en sortie.

5.2.4 La fonction ecriture.m

Appel :

```
ecriture(fich,ignore);
```

Rôle :

Elle écrit les données en sortie dans le fichier spécifié par l'utilisateur.

Commentaires :

Les variables de sortie du programme sont contenues dans la variable globale *M*. *M* est une matrice dont les colonnes sont, dans l'ordre : le temps, le champ magnétique, le signal en phase, le signal hors-phase, la phase et le signal démodulé total.

Cette fonction vérifie si le fichier de sortie spécifié par l'utilisateur existe ou pas. S'il existe, on demande à l'utilisateur s'il veut l'écraser, afin de protéger les anciennes données.

5.3 La fonction raffinage.m

Appel :

```
raffinage;
```

Rôle :

Elle concentre l'ensemble des post-traitements, c'est-à-dire traitements sur le signal démodulé, entrés par l'utilisateur par l'intermédiaire de l'interface.

Commentaires :

Elle est activée par l'utilisateur grâce au bouton **'Raffinage'** de l'interface.

Elle évalue tout d'abord les variables saisies par l'utilisateur dans l'interface. Ensuite, elle effectue les filtrages spécifiés sur le signal démodulé, de la même manière que l'on effectue le préfiltrage dans traitement.

Enfin, nous nous devons de mettre en évidence un problème qui se pose au vu des données en sortie. Sachant que chacun des 5 vecteurs en sortie fait *a priori* la même taille que les vecteurs en entrée, le fichier de sortie fera 5/3 du fichier en entrée, ce qui est dommage, puisque le principe de la détection synchrone est d'extraire une information d'un plus gros groupe d'informations. Ainsi, pour rendre les donnnées de sortie plus maniables, l'utilisateur a la possibilité de réduire les données en sortie d'un facteur *REDUC*.

raffinage fait alors appel à reduc_points.

5.3.1 La fonction reduc_points.m

Appel :

```
mat_reduite=reduc_points(mat,p);
```

Rôle : Elle permet de réduire le nombre de points dans les vecteurs de sortie.

Commentaires :

Le principe de cette fonction rejoint celui de lissage_magn. Elle effectue un moyennage par paquets de p points sur les différents vecteurs colonnes à réduire de *mat*, contenant les résultats. Enfin ,elle stocke les vecteurs réduits, par colonnes, dans *mat_ reduite*.

La fin de raffinage est dédiée au tracé de graphes, au calcul et tracé de transformées de Fourier, et enfin à l'écriture des données en sortie par ecriture (cf 5.2.4).

Troisième partie

Validation expérimentale du programme

Chapitre 6

Tests sur des données fabriquées

6.1 Définition des données

Avant de tester notre programme sur des données bruitées provenant d'expériences du L.N.C.M.P., nous avons décidé de le soumettre à toute une série de tests sur des données que nous avons fabriquées nous même. Nous avons choisi de construire une porteuse de fréquence 1 KHz échantillonnée à 100 KHz, le signal ayant une durée d'une seconde :

```
fs=100000;          %frequence d'echantillonnage
f0=1000;            %frequence de la porteuse
x=[1/fs:1/fs:1];    %vecteur temps
y=sin(2*pi*f0*x);   %porteuse
Vi=x.^2;            %signal a recuperer
v=Vi.*y;            %signal module en amplitude
```

Nous avons écrit ces données dans un fichier *essai.dat* et nous avons pu tester les différents résultats que nous obtenons avec notre programme.

6.2 Validation de la démodulation

Voilà les courbes que nous avons obtenues lors de la détection synchrone avec une fréquence de coupure de 500 Hz :

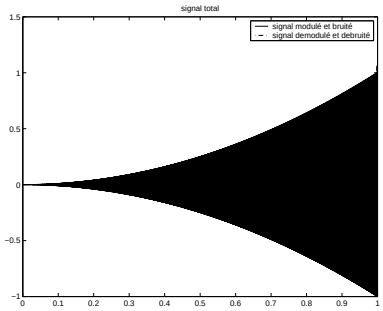


Fig. 17 : Courbes des signaux modulés et démodulés.

Lorsqu'on zoome sur quelques periodes du signal modulé, on s'aperçoit que la démodulation a parfaitement fonctionné puisque le signal démodulé correspond bien à l'enveloppe du signal modulé :

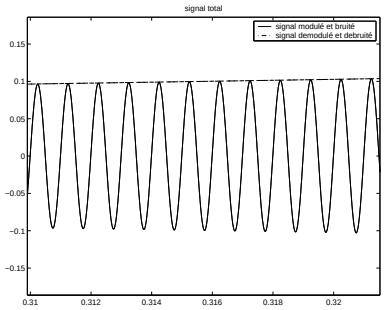


Fig. 18 : Zoom sur les courbes des signaux modulés et démodulés.

On peut vérifier avec les graphes des transformées de Fourier que nous proposons en option que tout s'est déroulé correctement :

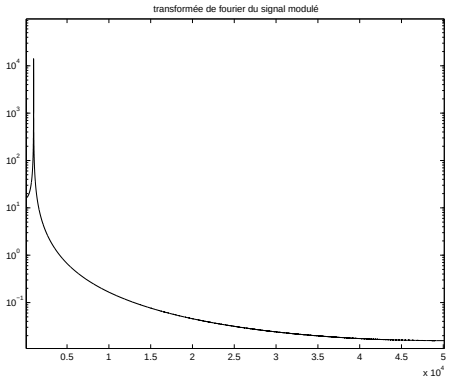


Fig. 19 : FFT du signal modulé.

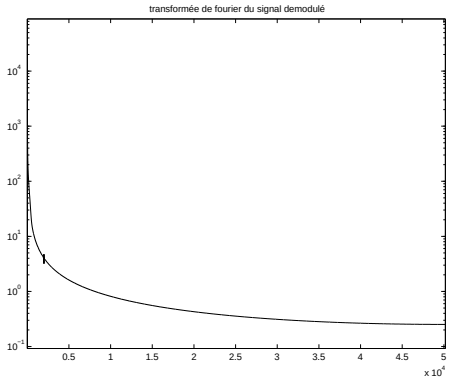


Fig. 20 : FFT du signal démodulé.

Si on zoome sur les fréquences peu élevées, on voit bien le pic correspondant à la fréquence de la porteuse (1000 Hz). En revanche, le signal démodulé ne contient que des fréquences très proches de 0 Hz, puisqu'il ne reste plus que le signal correspondant à la fonction carré :

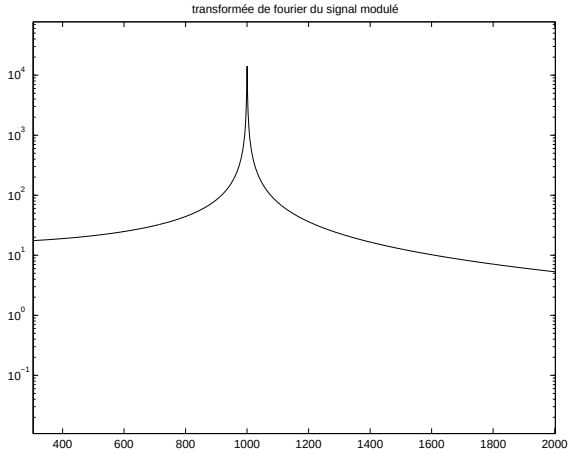


Fig. 21 : zoom sur la fréquence. 1000 Hz du signal modulé

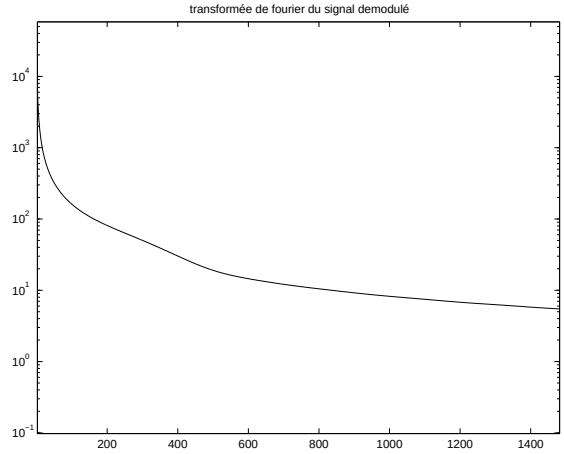


Fig. 22 : zoom sur les fréquences proches. de 0 pour le signal démodulé

6.3 Tests sur la phase

La phase est un paramètre important pour l'interprétation des résultats expérimentaux par les chercheurs du L.N.C.M.P.. C'est pourquoi il faut être sûr de sa valeur et de son signe. Commençons par observer la phase dans le cas où comme précédemment, elle est théoriquement nulle, c'est à dire que le signal modulé et la porteuse sont en phase (physiquement, le materiau sera purement résistif).

Les graphes suivants montrent en fonction du temps : (a) le signal en phase, (b) le signal hors phase, (c) le signal total et enfin (d) la phase elle même :

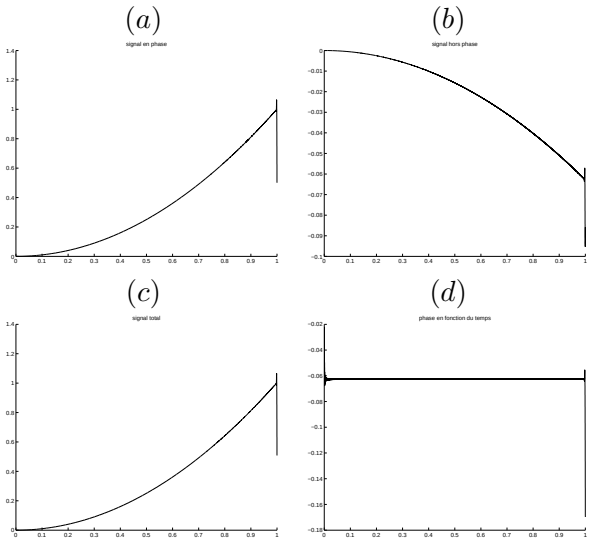


Fig. 23 : Tests pour $\phi = 0$.

On voit bien que la quasi-totalité de l'information est contenue dans le signal en phase. Notre programme trouve une phase de -0.05 radian, ce qui explique que le signal hors phase ne soit pas complètement égal à zéro. Cette petite erreur peut s'expliquer facilement en se rappelant que notre programme décale et ajuste la porteuse pour créer les sinus et cosinus nécessaires à la multiplication. Or ce décalage est valable à 1 point près. Une erreur de 1 point correspond à $\frac{1}{100} \times 2\pi = 0.0628$ rad dans l'exemple choisi puisqu'une periode est tracée sur 100 points. Pour la plupart des données que le programme aura à traiter, une periode de la porteuse contiendra environ 100 points, c'est pourquoi on peut conclure que **la valeur de la phase est bonne au dixième de radian près**.

Etudions à présent le signe de la phase. Pour cela, définissons la porteuse non plus comme

```
y=sin(2*pi*f0*x);
```

mais plutôt comme

```
y=sin(2*pi*f0*x-pi/6);
```

On s'attend alors à récupérer une phase constante positive égale à $\pi/6$.

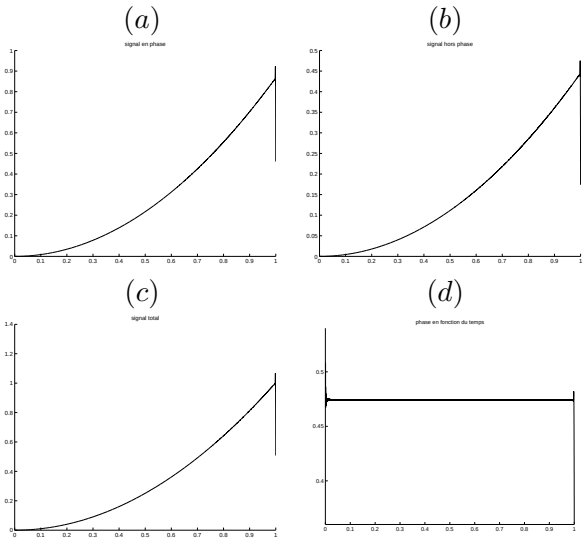


Fig. 24 : Tests pour $\phi = \frac{\pi}{6}$.

Les résultats sont conformes à ce que l'on attendait. On remarque que les signaux en phase et hors phase sont positifs (car $\sin(\frac{\pi}{6}) > 0$ et $\cos(\frac{\pi}{6}) > 0$. La valeur maximale du signal hors phase est environ 0.5 ce qui correspond bien à $\sin(\frac{\pi}{6})$. On vérifie de même que le maximum du signal en phase correspond à $\cos(\frac{\pi}{6})$.

Si on définit la porteuse comme

```
y=sin(2*pi*f0*x+pi/3);
```

on obtient les résultats suivants :

La phase est bien négative à présent.

Enfin le dernier test que l'on peut réaliser est de poser une phase non constante. On définit alors :

```
y=sin(2*pi*f0x+pi*x);
```

On s'attend donc à avoir une phase linéaire en fonction du temps et de pente égale à -1 :

Les signaux en phase et hors phase sont moins intuitifs à saisir mais on obtient toujours bien le carré en signal total. Le décrochement de la phase à $-\frac{\pi}{2}$ s'explique par le fait que le calcul de la phase est obtenue par l'application de la fonction arctangente au rapport $\frac{V_x}{V_y}$. Or arctangente est à valeur dans $\left[-\frac{\pi}{2}; \frac{\pi}{2}\right]$. Il ne faudra donc pas s'étonner de voir par la suite de telles discontinuités.

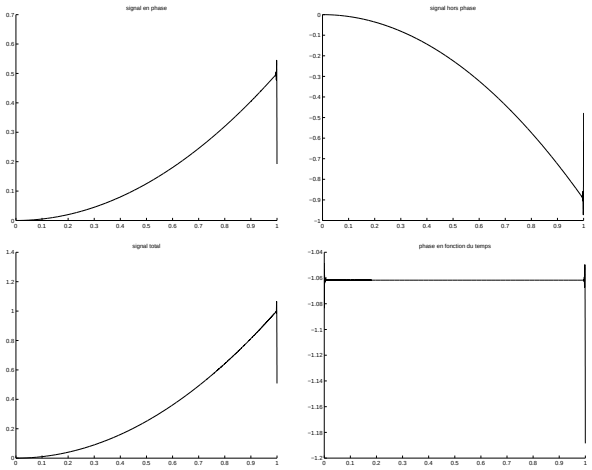


Fig. 25 : Tests pour $\phi = -\frac{\pi}{3}$.

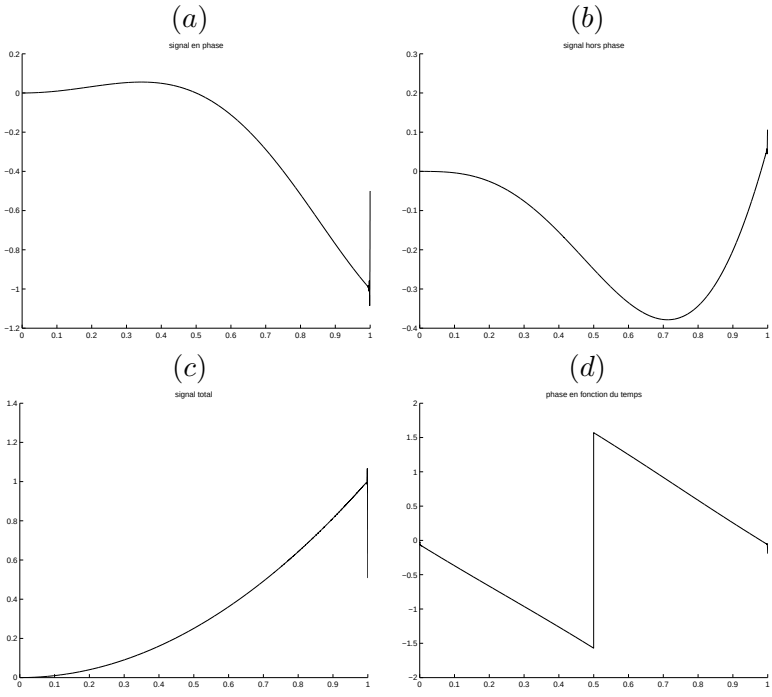


Fig. 26 : Tests pour ϕ non constant.

Chapitre 7

Tests sur des données provenant du L.N.C.M.P.

7.1 Rareté des données

Il faut tout d'abord savoir que le laboratoire n'effectue pas des "tirs" tous les jours et que parfois, la préparation d'un échantillon peut prendre plusieurs semaines. De plus, certains tirs sont manqués car les précautions pour atténuer le bruit n'ont pas été assez grandes et les données issues de l'expérience sont alors inexploitables. Les réactions du matériau étant souvent d'origine quantique, la minutie de l'expérience est primordiale.

Nous n'avons donc disposé que de 2 fichiers de données exploitables et qui ont servi à valider le bon fonctionnement de notre logiciel. Nous ne ferons bien évidemment aucune conclusion sur les propriétés physiques des matériaux car nous n'avons pas la formation de physicien nécessaire et nous laissons cela aux chercheurs du L.N.C.M.P.. Observons les résultats que nous avons obtenus.

7.2 Premier fichier de données : *detsyn.dat*

Tout d'abord, il faut signaler que nous ne possédons pour ces données aucun résultat obtenu avec la détection analogique mais nous avons pu constaté visuellement au laboratoire que les résultats que nous obtenons avec la détection numérique sont rigoureusement identiques.

7.2.1 Le signal à démoduler

Le signal que nous avons eu à démoduler comporte les paramètres suivants :

- la fréquence de la porteuse ($f0$ = 5050 Hz)
- la fréquence d'échantillonnage (fs = 76923 Hz)

– la durée de la mesure ($t = 1.3$ s).

Voila le signal que nous avions à démoduler :

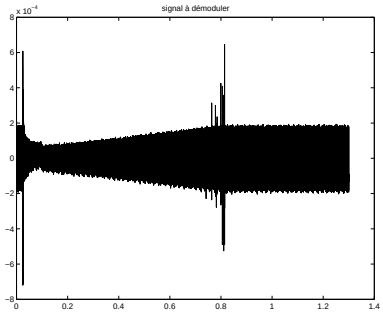


Fig. 27 : Signal à démoduler.

On voit que ce signal est très bruité et on va donc tester la démodulaton avec nos options de lissage du champ magnétique, de préfiltrage de raffinage.

7.2.2 Résultats de la détection synchrone numérique

Les graphes suivants montrent le signal total en fonction du champ magnétique : (a) sans préfiltrage, (b) avec un filtre passe-bande autour de 5050 Hz, (c) avec préfiltrage et lissage du champ magnétique et (d) avec le raffinage (filtre passe-bas et diminution du nombre de points par 10) :

On peut observer l'effet du préfiltrage sur les transformées de Fourier :

Nous voyons bien qu'une bonne partie des fréquences parasites ont été éliminées.

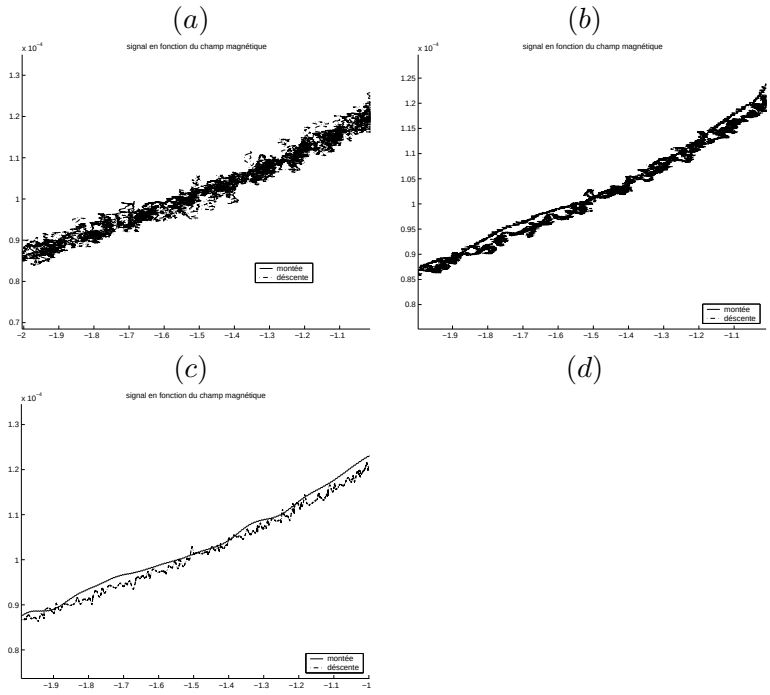


Fig. 28 : Utilisation des options de débruitage lors de la détection.

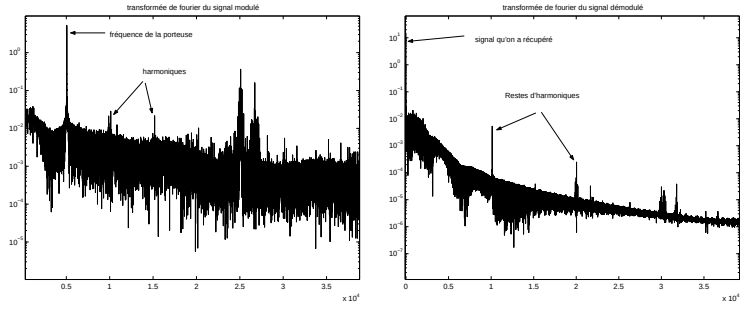


Fig. 29 : Transformées de Fourier sans préfiltrage.

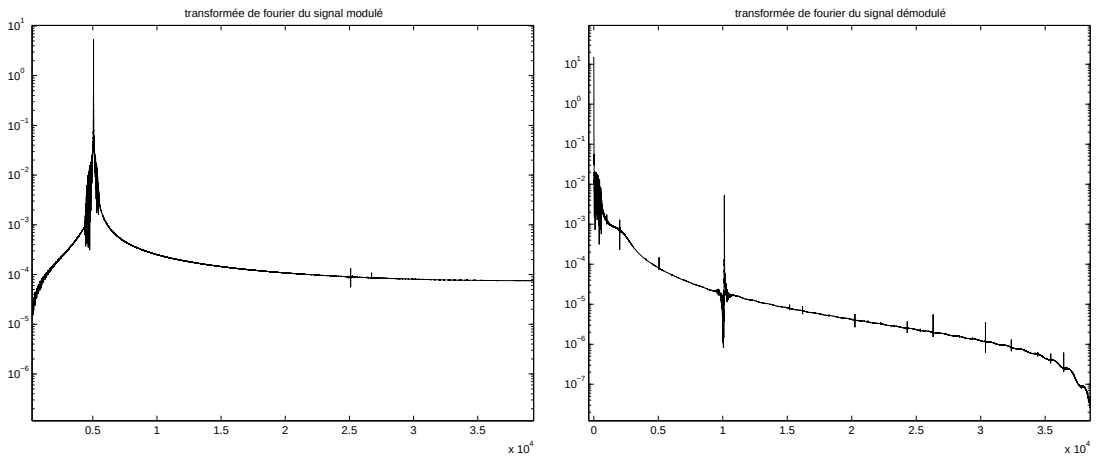


Fig. 30 : Transformées de Fourier avec préfiltrage.

7.3 Deuxième fichier de données : *VsdH01.dat*

Pour ces données-là, on possède les résultats obtenus avec la détection analogique et on est obligé de constater la superiorité de nos résultats numériques.

7.3.1 Le signal à démoduler

Le signal que nous avons eu à démoduler comporte les paramètres suivants :

- la fréquence de la porteuse ($f0 = 1000$ Hz)
- la fréquence d'échantillonnage ($fs = 50000$ Hz)
- la durée de la mesure ($t = 1$ s)

Voila le signal que nous avions à démoduler :

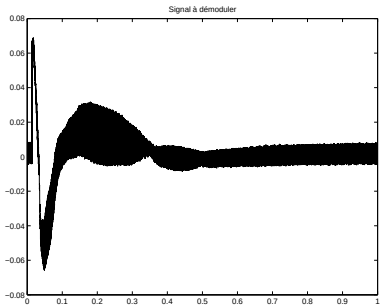


Fig. 31 : Signal à démoduler.

On peut tout d'abord remarquer une anomalie puisque ce signal n'est pas centré en zéro alors qu'il devrait logiquement l'être. Ce décrochage est caractéristique de certaines experiences et il serait un peu trop compliqué d'en expliquer la cause physique.

7.3.2 Résultats de la détection synchrone analogique

La detection analogique donne les courbes suivantes avec dans l'ordre (a) le signal en phase, (b) le signal hors phase, (c) le signal total en fonction du champ magnétique et enfin (d) la phase en fonction du temps :

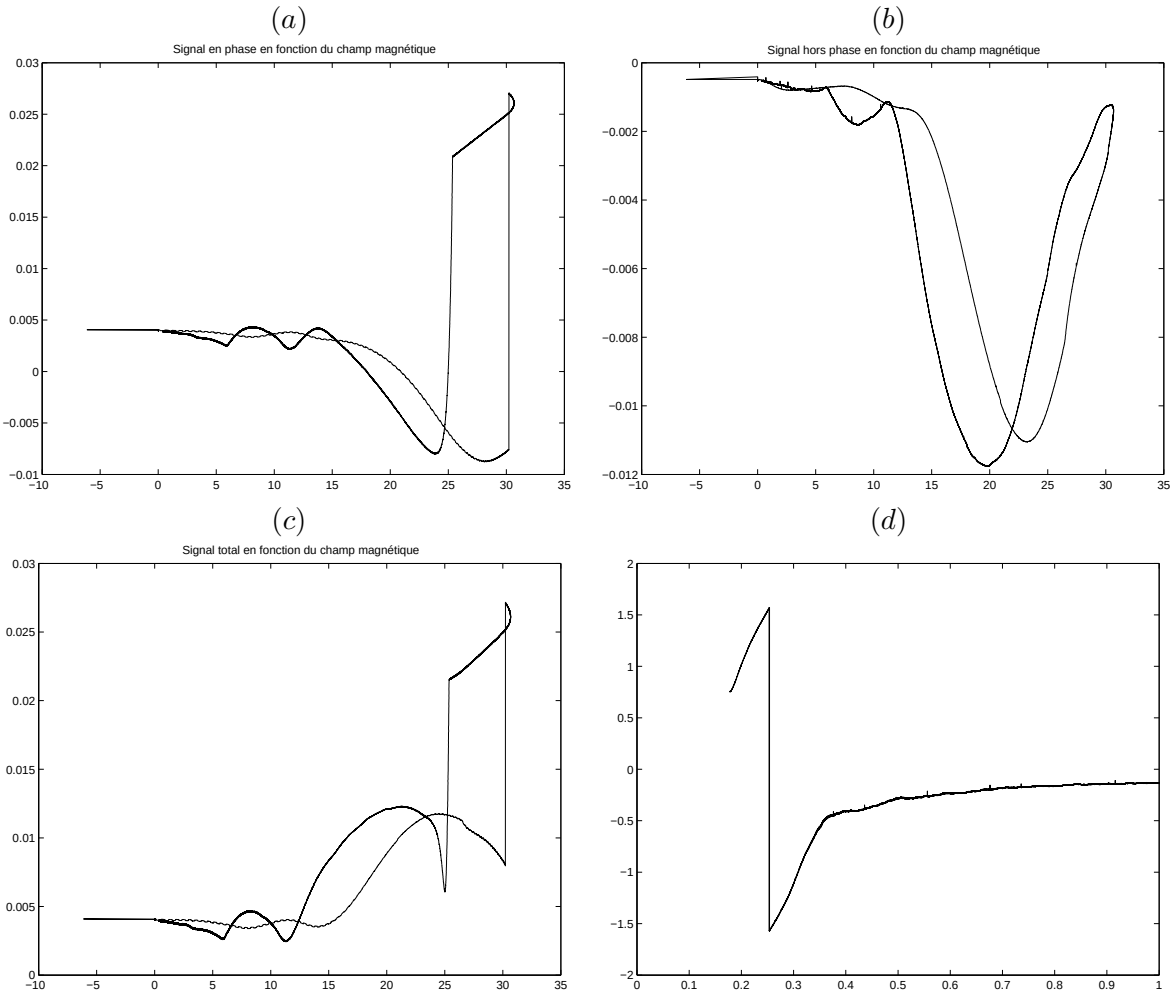


Fig. 32 : Résultats obtenus avec la détection analogique.

On voit que le signal mesuré lors de la montée ne se superpose pas exactement avec celui de la descente alors que les chercheurs auraient attendus une superposition. La phase présente 3 discontinuités dont 1 (celle du milieu) ne correspond pas à un passage de $\frac{\pi}{2}$ ou $-\frac{\pi}{2}$, mais plutôt une discontinuité liée soit à une erreur de la détection syncrone analogique, soit à une caractéristique du materiau. Il s'avère qu'avec notre détection numérique, nous trouvons de meilleurs résultats ce qui semble indiquer une défaillance des appareils électroniques de détection synchrone.

7.3.3 Résultats de la détection synchrone numérique

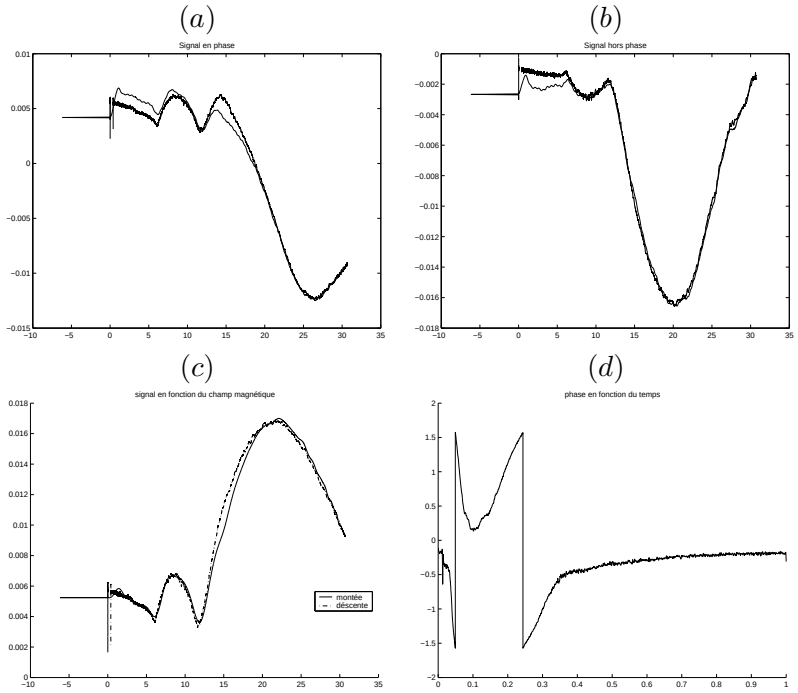


Fig. 33 : Résultats obtenus avec la détection numérique.

Les parties correspondant à la montée et à la descente du champ magnétique sont maintenant superposées et la phase ne présente plus de discontinuité anormale. D'autre part on peut vérifier visuellement que la démodulation s'est bien effectuée. Pour cela, il faut enlever la composante continue indésirable du signal modulé et vérifier si l'enveloppe correspond bien au signal démodulé. On applique donc un filtre passe-bande autour de la fréquence de la porteuse (1000 Hz) en préfiltrage :

On voit que maintenant le signal modulé est centré et que son enveloppe correspond bien au le signal démodulé.

7.3.4 Comparaisons des deux résultats et conclusion

Il est intéressant de superposer les graphes des signaux en fonction du temps ((a) signal en phase (b) signal hors phase et (c) phase) pour voir à quel moment la détection analogique a eu une défaillance :

On voit bien sur (a) et (c) que le décrochage a lieu entre environ 0.1 sec et 0.2 sec, c'est à dire quand la phase est au plus bas.

On peut donc être satisfait de notre détection synchrone qui semble être fiable et plus robuste que la détection analogique.

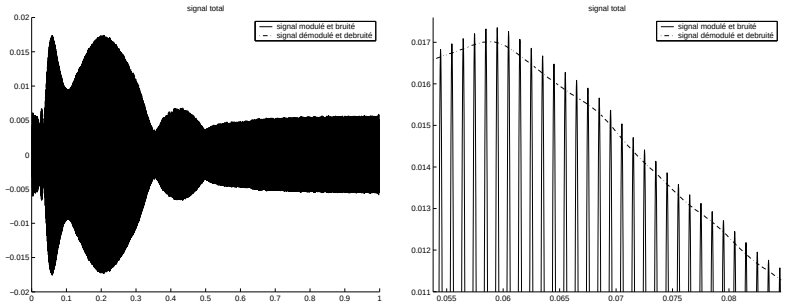


Fig. 34 : Signaux modulés et démodulés (avec l'option de préfiltrage).

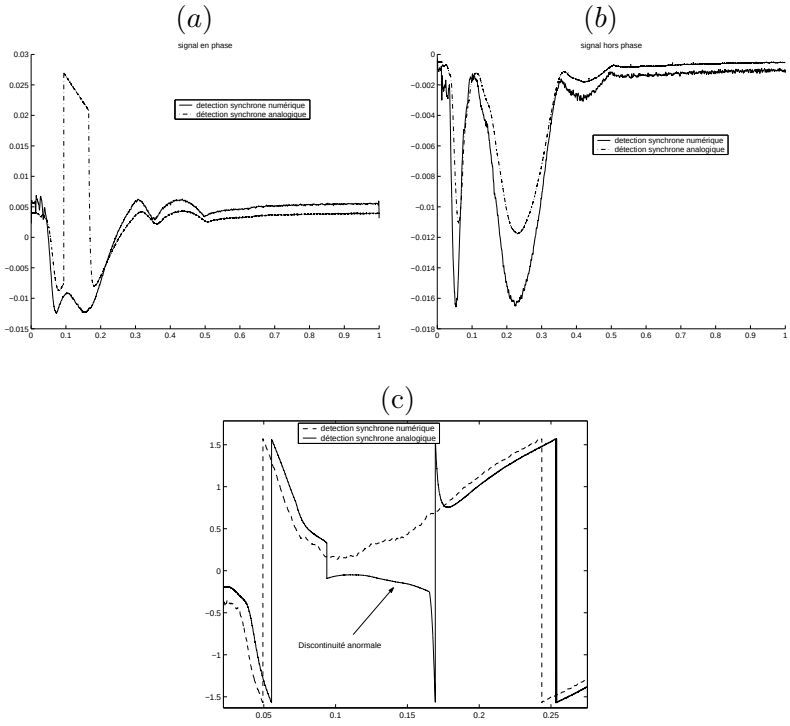


Fig. 35 : Comparaison des résultats de la détection numérique et analogique.

Chapitre 8

Etude des performances de notre logiciel

8.1 Etude de l'influence de la précision des paramètres en entrée sur les résultats

Il est utile à l'utilisateur de connaître la précision à apporter pour les données en entrée, notamment pour celles qui ne sont pas forcément connues exactement. On s'intéressera en particulier ici à la fréquence de la porteuse et à la fréquence d'échantillonnage.

8.1.1 Fréquence de la porteuse $f0$

On va juger de l'influence de sa variation dans la sous-fonction centrage. Elle intervient dans le cadre de la détermination du nombre de points décrivant une période de la porteuse. On rappelle qu'on pose :

```
points_periode=floor(freq_ech/f0);% frequence d echantillonnage
                                  % sur frequence de la porteuse
```

Considérons maintenant une variation ϵ sur cette fréquence. On a alors $points_{periode} = \frac{f_s}{f0+\epsilon}$. En supposant ϵ petit, on peut faire un développement limité et on obtient $points_{periode} \simeq \frac{f_s}{f0} \times (1+\epsilon)$. Or on effectue des recherches du maximum sur un intervalle de $\pm\frac{1}{5}\,periode$, recherches espacées de dix périodes. On tolère donc une erreur de $\frac{1}{50}^{\text{ème}}$ sur le nombre de points d'une période. Ceci équivaut à peu près à tolérance de 2% sur la fréquence entrée.

Ceci suffit à conserver une valeur exacte pour l'amplitude et la composante continue, mais le décalage entre les deux références (cf 5.2.2) pâtira directement de toute erreur sur la fréquence de la porteuse.

Les autres emplois de $f0$ sont plus secondaires. On peut citer par exemple l'établissement du degré de base des filtres.

8.1.2 Fréquence d'échantillonnage f_s

On peut faire la même analyse que précedemment pour le calcul de l'amplitude et de la composante continue, et on obtiendra $points_{periode} = \frac{f_s}{f0} \times (1 + \epsilon)$. Pour ce calcul on tolère là aussi une erreur de 2% sur la fréquence entrée.

Toutefois, pour le décalage entre les deux références, on commettra ici encore une erreur irréversible, dès lors que f_s ne sera plus exacte.

Enfin, f_s intervient directement au dénominateur dans l'ensemble des calculs pour établir les coefficients des différents filtrages, et donc les filtrages induits ne seront plus ceux souhaités en cas d'erreur sur f_s.

C'est pourquoi il semble important de conserver une précision relative la plus faible possible sur $f0$ et plus encore sur f_s.

Nous allons maintenant étudier rapidement le temps de calcul et le nombre d'opérations de notre programme pour que l'utilisateur puisse éviter d'utiliser s'il le désire les options qui demandent beaucoup de calculs.

8.2 Etude du nombre d'opérations

Tout d'abord, il faut savoir que nous n'avions pas une grande contrainte d'optimisation du code de notre programme pour la simple raison que les données étant relativement rares, le temps consacré par les chercheurs à l'analyse des données peut être important. Ce n'aurait donc pas été un gros problème si la durée de traitement avait été de quelques minutes.

Cependant, nous avons tout de même essayé de minimiser les calculs pour rendre l'utilisation du logiciel plus agréable. Nous avons tout d'abord évité les boucles qui sous Matlab prennent un temps considérable. Il faut noter cependant qu'un des inconvénients de ce refus absolu des boucles est que le code (en particulier pour le calcul du lissage du champ magnétique ou pour la réduction du nombre de points) perd un peu de sa clarté.

Pour un fichier contenant 100000 points, voilà les résultats que nous obtenons :

– dans traitement.m :

Tâche	Nombre d'opérations
lissage (facteur1=100)	695 431
centrage	200 523
multiplication par un cosinus	199 990
multiplication par un sinus	199 990
calcul des coefficients du filtre	222 126
filtrage de yphase	24 998 751
filtrage de yhorsphase	24 998 751
calcul de la phase	199 990
calcul du module du signal	399 980
FFT du signal total	1 999 300 000
tracé des signaux modulés et démodulés	1 069
ecriture dans le fichier	492

– dans raffinage.m :

Tâche	Nombre d'opérations
réduction du nombre de points (facteur2=100)	605 993
FFT du signal réduit	175 536

Il apparaît tout de suite que deux type de tâches sont très gourmandes en calcul : le filtrage et le calcul de la FFT.

Sous Matlab, un filtrage de type $x \mapsto y = h * x$ n'est pas calculé par convolution car le calcul de tous les termes

$$y_n = \sum_{k \in \mathbb{Z}} h_n y_n$$

prendrait alors $2 \times N^2$ opérations. Les filtres que nous utilisons étant à réponse impulsionnelle finie et de la forme

$$h = \sum_{n=0}^{r} b_n \delta_n \quad \text{(où r est le degré du filtre)},$$

Matlab utilise plutôt un calcul direct $y_n = b_0 x_n + b_1 x_{n-1} + ... + b_r x_{n-r}$ qui nécessite lui $2 \times r$ opérations. Donc au total, on obtient $2 \times r \times N$ opérations si N est la longueur du vecteur à filtrer. Typiquement, dans notre programme, les filtres sont au moins de degré 100, et les vecteurs environ de taille 100 000 (comme dans notre exemple), ce qui nous fait donc au minimum 20 000 000 opérations (à comparer avec les 24 998 751 que l'on obtient experimentalement).

Il faudra donc réfléchir avant de demander l'option de préfiltrage. En revanche les deux filtrages que l'on effectue pour obtenir le signal en phase et hors phase sont inévitables.

Il en est de même pour les transformées de Fourier qu'il faut demander seulement si on en a l'utilité car l'algorithme de calcul d'une FFT prend quant à lui au mieux $1.5 \times N \times \log_2 N$ (dans le cas où $N = 2^p$) soit dans notre exemple, théoriquement 2 491 400 opérations pour le vecteur de longueur 100 000 et 14 949 opérations pour le vecteur réduit de longueur 1000. Mais en réalité, on s'aperçoit qu'il prend bien plus d'opérations.

Les autres tâches sont négligeables en terme de nombre d'opérations et l'utilisateur n'a pas à hésiter à prendre l'option de lissage du champ magnétique ou celle de réduction des points car les calculs qui leur sont affectés sont seulement de l'ordre de la longueur des vecteurs.

Mais le nombre d'opérations n'est pas toujours corrélé au temps d'exécution en particulier quand Matlab trace des graphes ou lit et écrit dans des fichiers. Il est donc intéressant pour l'utilisateur de savoir quelles parties de notre programme prennent le plus de temps en exécution.

8.3 Etude du temps de calcul

Cette petite étude du temps de calcul a été réalisée grâce aux commandes *tic* et *toc* de Matlab qui respectivement déclenchent et arrêtent un chronomètre. Ces temps de calcul ont été calculés sur un Pentium II cadencé à 300 Mhz, ce qui correspond à peu près à l'ordinateur sur lequel sera réalisé les détections synchrones au L.N.C.M.P.. Pour les dernières générations de PC cadencées à plus d'1 GHz, il est bien évident que le temps de calcul sera bien moindre mais les proportions observées ci-après resteront les mêmes.

Voici le tableau résumant les résultats de cette étude pour l'exécution d'un traitement, sachant qu'il ne fait apparaître que les tâches prenant des temps significatifs (supérieurs à un dixième de seconde) :

Tâche	temps (en seconde)
sauvegarde des données dans rappel.mat	3.02
chargement des données du fichier ascii	3.57
lissage du champ magnétique	0.50
préfiltrage (avec un passe-bande)	4.12
filtrage de yphase	1.65
filtrage de yhorsphase	1.86
FFT du signal total	0.54
tracé du graphe non optionnel	5.33
tracé des graphes optionnels	6.59
écriture ds le fichier	6.37

Sur les 35 secondes de calculs du traitement, on voit qu'une part non négligeable du temps (plus de 10 secondes) est passée au préfiltrage et au tracé

des graphes optionnels. On se rend aussi compte de la lenteur d'accès aux fichiers que cela soit en lecture mais surtout en écriture, à cause de la taille importante des fichiers (plusieurs Mo). Pour l'écriture de certains fichiers particulièrement volumineux, on peut arriver jusqu'à 20 secondes passées à l'écriture. L'utilisateur n'hésitera donc pas à utiliser l'option de réduction du nombre de points en sortie. Il accélérera ainsi de même le tracé des graphes qui peut lui aussi prendre beaucoup de temps surtout pour le graphe non optionnel (jusqu'à 12 secondes) qui contient beaucoup d'informations.

Conclusion

Ce projet aura été l'occasion pour nous d'approfondir nos capacités de programmation MATLAB, mais on peut voir au delà, pour affirmer qu'il aura été riche d'enseignements pour nous, en particulier au niveau de l'évolution de la réflexion dans un groupe et de la découverte d'un nouvel environnement. Par ailleurs, nous avons pu nous rendre compte des interférences dues aux divergences lexicographiques, entre une population de physiciens et nous, dont la formation a été plus axée sur les mathématiques. Ceci peut sembler un point de détail, toutefois il semble dommageable que des personnes appelées à travailler ensemble soient ralenties par ce genre de difficultés.

Enfin, nous devons souligner une nouvelle fois la qualité de l'encadrement qui, à la différence de celui d'autres binômes de notre classe, a su être secourable sans jamais devenir étouffant. Ceci étant écrit en toute honnêteté.

Bibliographie

[1] MATLAB : *Signal prorocessing toolbox (user's guide).*

[2] J-P. Delmas : *Eléments de théorie du signal : Les Signaux déterministes.*

[3] J. Max : *Méthodes et techniques de traitement du signal et applications aux mesures physiques. TOME PREMIER : Principes et appareillages de traitement en temps réel.*

[4] Mokhtari - Mesbah : *Apprendre et maîtriser Matlab.*

DOCUMENTS INTERNES :

[5] Dpt GP de l'INSAT. J-P. Ulmet : *Traitement du signal-Bruit de fond.*

[6] Dpt GP de l'INSAT. B. Raquet : *Détection synchrone.*

[7] Dpt GMM de l'INSAT. A. Bendali, P. Guillaume et J-P. Vila : *Cours d'analyse fonctionnelle et traitement du signal.*

[8] Dpt GEII IUT Bordeaux 1. G. Couturier : *Bruit en électronique et détection synchrone.*

[9] Groupe ESIEE. O. Français : *Détection synchrone.*

[10] L.N.C.M.P. : *Rapport d'activité 1999-2000.*

Annexes : Programmes Matlab