Traitement numérique du signal

Julien Salomon

23 juillet 2010

Introduction

Objectifs du cours

Le but principal de ce cours est de mieux comprendre les techniques, méthodes et outils mathématiques utilisés dans ce qu'on appelle habituellement le traitement numérique du signal.
Ce faisant, il doit permettre de mieux comprendre les différents problèmes scientifiques rencontrés et plus généralement de familiariser le lecteur à un domaine qui occupe de nos jours une place importante dans la plupart des objets techniques de la vie courante.
Mise à part le cours d'intégration de L1 et l'analyse de Fourier, peu de pré-requis sont nécessaires pour l'aborder. Dans chaque chapitre, sauf quelques rares exceptions, on reste centré autour d'une étape de la transmission d'un signal numérique. Pour autant, les outils qui y sont présentés ont souvent une portée qui dépasse le simple cadre du traitement numérique du signal, si bien que le cours a aussi vocation à ouverture scientifique et culturelle.

Chaîne de transmission numérique

Prenons l'exemple de la transmission d'une conversation téléphonique, et voyons quelles sont les différentes procédures que subit le signal entre son émission et sa réception.

Conversion analogique-numérique

Lorsque l'on parle dans le microphone d'un téléphone, les vibrations acoustiques sont transformées en une tension oscillante par l'intermédiaire d'une membrane et d'un électro-aimant qui convertit ainsi le signal mécanique en signal électrique. C'est à ce moment qu'intervient la conversion analogique-numérique, c'est-à-dire le passage du continu au discret, de l'analogique au numérique, de $\mathbb{R} \to \mathbb{R}$ à $\mathbb{Z} \to \mathbb{Z}$ en quelque sorte. Le continu -$\mathbb{R}$- se trouve dans deux caractéristiques du signal électrique : le temps d'une part et les valeurs prises par la tension. Pour obtenir un signal complètement numérique, la tension est donc discrétisée en temps et en valeur. Concrètement, cela revient à prendre une série de photos instantanées des valeurs de la tension, puis de projeter les valeurs obtenus sur une grille fixe. Dans le vocabulaire du traitement numérique du signal, ces deux étapes sont respectivement appelées *échantillonnage* et *quantification*. L'ensemble de ces deux étapes constitue la conversion analogique-numérique, souvent notée C.A.N.

Codage et compression

Revenons à notre exemple de conversation téléphonique. Les normes suivies dans les télécommunications sont fixées au niveau international par l'I.U.T, l'union internationale de télécommunications, c'est-à-dire un important (en taille et en influence) groupement d'ingénieurs et d'experts des télécommunication. Dans notre exemple, le signal analogique (la tension issue de la conversion signal mécanique/ signal électrique) est échantillonné à 8kHz et quantifié sur 8 bits. Cela signifie que l'on relève la valeur de la tension 8000 fois par seconde, et que la valeur obtenue est remplacée par une valeur choisie sur une grille en comportant $256 = 2^8$. Si l'on voulait transmettre telle quelle la liste de 0 et de 1 obtenue après ces deux étapes, il faudrait donc disposer d'un canal de transmission de débit 64kBits par seconde.
Au début des années 50, on pensait que ce débit était très peu compressible et qu'il fallait concevoir des dispositifs de transmission supportant de tels débits. A l'heure actuelle, on transmet correctement une conversation

téléphonique à l'aide d'un canal de débit de 4kBits par seconde. Comment a-t-on fait pour réduire autant (plus de 10 fois !) le volume d'informations à transmettre ? On a fait appel à des techniques de compression. L'ensemble de ces méthodes, appelées *codage source* développées à partir des années 50, peut être divisé en deux grandes catégories : la compression avec et la compression sans perte. Un représentant célèbre de la première catégorie est le format *mp3*, et un représentant célèbre de la seconde catégorie est le logiciel *zip*. Toujours dans l'exemple de la conversation téléphonique, ces deux techniques sont utilisées pour obtenir le débit de 4kBits par seconde. Ce débit prend en compte un autre traitement qu'a subi le signal à transmettre, le *codage canal* qui, au contraire de la compression, ajoute des redondances pour permettre de corriger les erreurs éventuelles qui vont intervenir lors de la transmission. Signalons que c'est également ce type de techniques qui permet de lire correctement des *CD* ou des *DVD* rayés.

Modulation et transmission dans le canal

Le signal – codé – peut alors être envoyé dans le canal de transmission qui peut être selon le téléphone utilisé, un câble en cuivre (cas courant du téléphone fixe), une fibre optique, l'atmosphère (cas du téléphone portable), l'espace (cas des communications par satellite)... Le but est alors d'adapter les symboles numériques, i.e. la séquence de 0 et de 1, au canal de transmission choisi. Concrètement, si l'on souhaite transmettre 1 bit en Δt seconde, il s'agit de construire un signal analogique qui garde une caractéristique constante pendant Δt seconde. Toute cette étape est généralement désignée sous le terme de *modulation*. Dans le canal, le signal codé subit des altérations de nature très variées, que le codage canal va pouvoir corriger.

Réception et décodage

A la réception en sortie de canal, interviennent séquentiellement les opérations inverses de celles présentées ci-dessus. On effectue donc une démodulation, un décodage canal, un décodage source et finalement une conversion numérique-analogique.

L'ensemble de cette chaîne de transmission est reproduite dans la figure 1.

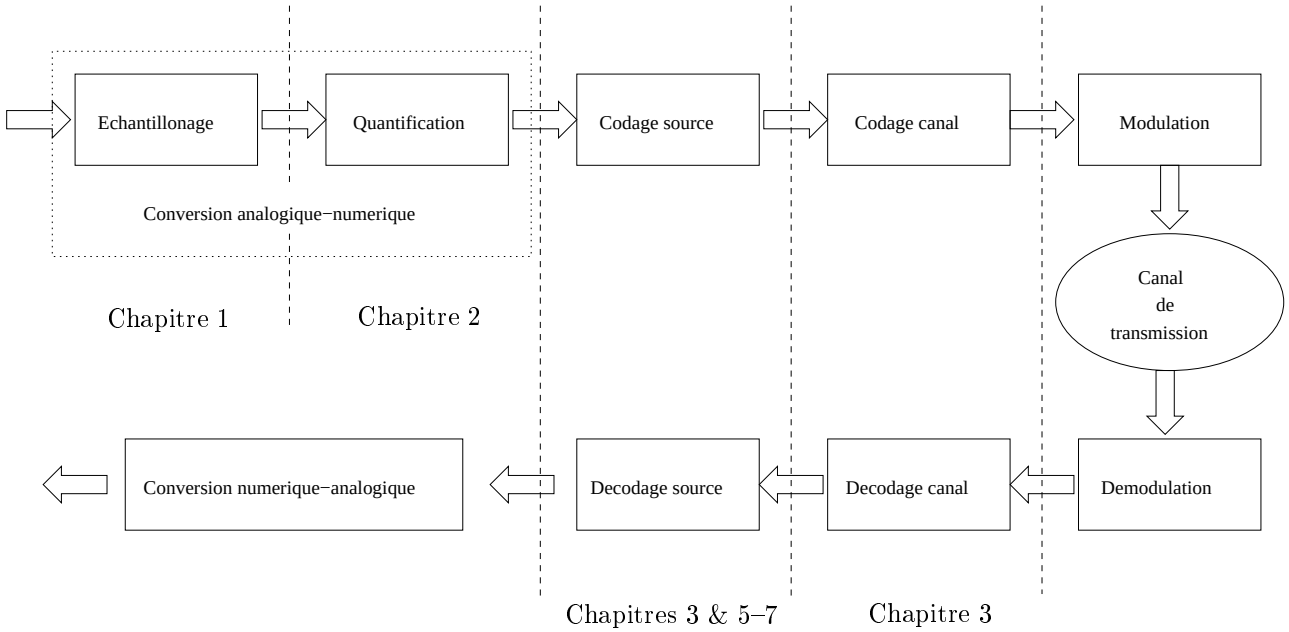

FIG. 1 – La chaîne de transmission du signal et chapitres correspondants

Plan du cours

Le plan du cours est basé sur les différentes étapes de la chaîne de transmission qui vient d'être décrite. Les chapitres 1 et 2 portent sur les conditions d'échantillonage et les techniques de quantification. Des techniques de codage source sans perte de codage canal sont présentées au chapitre 3. Les chapitres suivant permettent d'introduire la compression avec perte. Celle-ci repose presque systématiquement sur la décomposition du signal. Dans le cadre de ce cours, cette décomposition s'effectue suivant les composantes de Fourier du signal à transmettre. On commence donc par introduire la transformation de Fourier discrète ainsi qu'un algorithme très efficace permettant son calcul, la *transformée de Fourier rapide*, encore appelée FFT au chapitre 4. On aborde ensuite la notion de filtre numérique, outil indispensable à la décomposition d'un signal numérique au chapitre 5. La conception d'un filtre numérique est traitée au chapitre 6. Enfin, on présente le cadre général ainsi que les conditions d'une compression avec perte efficace au chapitre 7.
On parlera indifféremment de fonction ou de signal.

Le signe := signifie que l'égalité indiquée constitue une définition.

Table des matières

Chapitre 1

Formule de Shannon-Nyquist et échantillonnage

Dans ce chapitre, nous établissons un théorème qui permet, à partir de relevés discrets d'un signal, de le reconstruire entièrement. Ceci se fait bien entendu au prix de quelques hypothèses de régularité sur le signal. Avant d'énoncer et de démontrer le théorème en question, nous faisons quelques rappels sur la théorie de la Transformation de Fourier[1].

1.1 Transformées de Fourier des fonctions L^1

La transformée de Fourier désigne généralement une application qui agit sur des fonctions définies sur $\mathbb{R}^n$, avec $n \geq 0$. Pour autant, on peut définir les transformations dans des cadres beaucoup plus larges[2]. Nous verrons dans la suite de nombreux exemples de généralisation. Nous commençons par les fonctions de $L^1(\mathbb{R})$.

1.1.1 Définitions et notations

Introduisons tout d'abord quelques notations. On note :

$$L^1 := \left\{ f : \mathbb{R} \to \mathbb{C},\ \int_{\mathbb{R}} |f| < +\infty \right\}.$$

Pour ce qui nous concerne, nous désignerons par *signal stable* une fonction de L^1.
Étant donnée une partie A de $\mathbb{R}$, on note $\mathbb{1}_A$ sa fonction indicatrice, c'est-à-dire la fonction de $\mathbb{R}$ dans $\mathbb{R}$ qui vaut 1 sur la partie A et 0 ailleurs.
Une fonction est dite *à support borné* si elle est nulle en dehors d'une partie bornée de $\mathbb{R}$.
Venons-en à la définition qui nous intéresse.

Définition 1. *Soit $s \in L^1$. La transformée de Fourier de s, notée $\widehat{s}$ est définie sur $\mathbb{R}$ par la formule :*

$$\widehat{s}(\nu) := \int_{\mathbb{R}} s(t) \mathrm{e}^{-2i\pi\nu t} \mathrm{d}t.$$

Voici quelques exemples, dont les calculs peuvent être refaits à titre d'exercice et qui seront très utiles pour la suite.

1. Pour $T > 0$, la transformée de Fourier d'une fonction fenêtre, encore appelée *signal rectangulaire*, est donnée par :
$$rect_T(t) := \mathbb{1}_{[-T/2,T/2]}(t) \longrightarrow \widehat{rect_T}(\nu) := T\mathrm{sinc}(\pi\nu T) = T\frac{\sin(\pi\nu T)}{\pi\nu T}.$$

[1] Joseph Fourier (21 mars 1768, Auxerre - 16 mai 1830, Paris) est un mathématicien et physicien français, connu pour ses travaux sur la décomposition de fonctions périodiques en séries trigonométriques convergentes appelées séries de Fourier.
[2] En fait, les opérations algébriques nécessaires à sa définition sont celles d'un groupe.

2. Transformées de Fourier de Gaussiennes :

$$\mathrm{e}^{-\pi t^2} \longrightarrow \mathrm{e}^{-\pi \nu^2},$$

et par changement de variable :

$$\mathrm{e}^{-\alpha t^2} \longrightarrow \sqrt{\frac{\pi}{\alpha}} \mathrm{e}^{-\frac{\pi^2}{\alpha}\nu^2}.$$

1.1.2 Convolution

Une loi de composition est généralement associée à la transformée de Fourier, c'est le *produit de convolution*[3].

Définition 2. *Soit a, b deux fonctions de L^1. On définit le produit de convolution ou convolée* [4] *de a et b par la formule :*

$$a \star b := \int_{\mathbb{R}} a(t-u)b(u)\mathrm{d}u.$$

Cette définition a bien un sens, en effet, d'après le théorème de Tonelli, on a :

$$\int_{\mathbb{R}} \int_R |a(t-u)||b(u)|\mathrm{d}u\mathrm{d}t = \left(\int_{\mathbb{R}} |a|\right) . \left(\int_{\mathbb{R}} |b|\right),$$

et donc :

$$\int_{\mathbb{R}} |a(t-u)||b(u)|\mathrm{d}u < +\infty \qquad pp.,$$

par conséquent $t \mapsto \int_{\mathbb{R}} a(t-u)b(u)\mathrm{d}u$ est définie presque partout.
On vérifie aisément que cette loi de composition est commutative et associative. Le lien avec la transformée de Fourier est donné par le lemme suivant.

Lemme 1. *Soit a, b deux fonctions de L^1. On a :*

$$\widehat{a \star b} = \widehat{a}\widehat{b}.$$

Autrement dit, la transformée de Fourier change le produit de convolution en simple produit.

Démonstration. La fonction $a \star b$ étant intégrable, d'après le théorème de Fubini nous avons :

$$\begin{aligned} \int_{\mathbb{R}} \int_R a(t-u)b(u)\mathrm{d}u\mathrm{e}^{-2i\pi\nu t}\mathrm{d}t &= \int_{\mathbb{R}} b(u) \int_{\mathbb{R}} a(t-u)\mathrm{e}^{-2i\pi\nu(t-u)}\mathrm{d}t\ \mathrm{e}^{-2i\pi u}\mathrm{d}u \\ &= \widehat{a}\widehat{b}. \end{aligned}$$

□

1.1.3 Le lemme de Riemann-Lebesgue

Dans les démonstrations que nous verrons dans la suite, nous ferons souvent appel à des passages à la limite sous le signe intégral. Nous aurons en particulier besoin du résultat suivant, généralement appelé Lemme de Riemann-Lebesgue.

Lemme 2. *(Lemme de Riemann-Lebesgue) La transformée de Fourier d'un signal stable vérifie :*

$$\lim_{|\nu|\to+\infty} |\widehat{s}(\nu)| = 0.$$

Démonstration. On procéde en trois étapes.

[3] qui elle aussi peut être généralisée à des cadres plus larges que celui des fonctions de $L^1(\mathbb{R})$.
[4] Attention, ici le vocabulaire diverge entre physiciens et mathématiciens. En mathématiques, on parle de *convolée* de deux fonctions, alors que les physiciens emploient le terme *convoluée*.

1. Pour un signal rectangulaire, nous avons vu à la section 1.1.1 qu'il existe un $K > 0$ tel que :
$$|\widehat{s}(\nu)| \leq \frac{K}{|\nu|}.$$
Le résultat est donc vrai dans ce cas.
2. Ce résultat s'étend aux combinaisons linéaires finies de signaux rectangulaires, c'est-à-dire aux fonctions étagées.
3. Soit maintenant un signal stable s et un réel ν. Par densité des fonctions étagées dans L^1, nous savons qu'il existe une suite de fonctions étagées $(s_n)_{n\in\mathbb{N}}$ vérifiant :
$$\lim_n \int_{\mathbb{R}} |s_n - s| = 0.$$
D'autre part, nous avons :
$$|\widehat{s}(\nu) - \widehat{s_n}(\nu)| \leq \int_{\mathbb{R}} |s(t) - s_n(t)| \mathrm{d}t.$$
Puisque d'après les étapes précédentes :
$$|\widehat{s_n}(\nu)| \leq \frac{K_n}{|\nu|},$$
nous obtenons :
$$\begin{aligned} |\widehat{s}(\nu)| &\leq |\widehat{s_n}(\nu)| + \int_{\mathbb{R}} |s(t) - s_n(t)| \mathrm{d}t \\ &\leq \frac{K_n}{|\nu|} + \int_{\mathbb{R}} |s(t) - s_n(t)| \mathrm{d}t, \end{aligned}$$
qui peut être rendu arbitrairement petit, sous réserve que $|\nu|$ soit suffisamment grand.

□

1.1.4 La formule d'inversion

Les résultats précédents nous indiquent que la transformée de Fourier d'un signal stable tend vers 0. On peut également montrer qu'elle est uniformément continue et bornée. Par contre, elle ne constitue pas un signal stable. En effet, pour $T > 0$, nous avons :
$$\int_1^T \frac{|\sin(t)|}{t} \mathrm{d}t \geq \int_1^T \frac{\sin^2(t)}{t} \mathrm{d}t = \int_1^T \frac{1}{t} \mathrm{d}t - \int_1^T \frac{\cos(2t)}{2t}.$$
D'autre part :
$$\int_1^T \frac{\cos(2t) + 1}{2t} \mathrm{d}t = \int_1^T \frac{\cos^2(t)}{t} \mathrm{d}t,$$
et donc
$$\int_1^T \frac{|\sin(t)|}{t} \mathrm{d}t \geq \int_1^T \frac{1}{2t} - \int_1^T \frac{\cos(2t)}{2t} \mathrm{d}t,$$
ce dernier terme intégral étant convergent, la transformée de Fourier des signaux rectangulaires n'est pas dans L^1. Il faut donc le supposer, c'est-à-dire l'ajouter en hypothèse pour obtenir le résultat suivant.

Théorème 1. *(Formule d'inversion de la transformée de Fourier) Soit s un signal stable, tel que sa transformée de Fourier $\widehat{s}$ soit également stable. Alors, pour presque tout t :*
$$s(t) = \int_{\mathbb{R}} \widehat{s}(\nu) \mathrm{e}^{2i\pi\nu t} \mathrm{d}\nu.$$

Démonstration. On procède de nouveau en trois étapes.

1. Par un calcul simple, sans problème de convergence, on montre le résultat pour les fonctions $f_{\alpha,a}$ définies par :
$$f_{\alpha,a}(t) = \mathrm{e}^{-\alpha t^2 + at}.$$

2. On considère ensuite la fonction h_σ définie par :
$$h_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{t^2}{2\sigma^2}},$$
dont la transformée de Fourier est :
$$\widehat{h_\sigma}(\nu) = \mathrm{e}^{-2\pi^2\sigma^2\nu^2}.$$
Étant donné un signal stable s, la formule d'inversion est vraie pour $s \star h_\sigma$. En effet :
– on a successivement :
$$\begin{aligned} s\star h_\sigma(t) &= \int_{\mathbb{R}} s(u)h_\sigma(t-u)\mathrm{d}u \\ &= \int_{\mathbb{R}} s(u)\frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(t-u)^2}{2\sigma^2}du} \\ &= \int_{\mathbb{R}} s(u)h_\sigma(u)f_{\frac{1}{\sigma\sqrt{2}},\frac{u}{\sigma^2}}(t)\mathrm{e}^{-2\pi i\nu t}\mathrm{d}u, \end{aligned}$$
– de plus :
$$\begin{aligned} \widehat{s}(\nu)\widehat{h_\sigma(\nu)} &= \int_{\mathbb{R}}\int_{\mathbb{R}} s(u)h_\sigma(u)f_{\frac{1}{\sigma\sqrt{2}},\frac{u}{\sigma^2}}(t)\mathrm{e}^{-2\pi i\nu t}\mathrm{d}t\mathrm{d}u \\ &= \int_{\mathbb{R}} s(u)h_\sigma(u)\underbrace{\int_{\mathbb{R}} f_{\frac{1}{\sigma\sqrt{2}},\frac{u}{\sigma^2}}(t)\mathrm{e}^{-2\pi i\nu t}\mathrm{d}t}_{\widehat{f}_{\frac{1}{\sigma\sqrt{2}},\frac{u}{\sigma^2}}(\nu)}\mathrm{d}u, \end{aligned}$$
– donc :
$$\begin{aligned} \int_{\mathbb{R}}\widehat{s}(\nu)\widehat{h_\sigma}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu &= \int_{\mathbb{R}}\int_{\mathbb{R}} s(u)h_\sigma(u)\widehat{f}_{\frac{1}{\sigma\sqrt{2}},\frac{u}{\sigma^2}}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}u\mathrm{d}\nu \\ &= \int_{\mathbb{R}} s(u)h_\sigma(u)f_{\frac{1}{\sigma\sqrt{2}},\frac{u}{\sigma^2}}(t)\mathrm{d}u \\ &= s\star h_\sigma(t), \end{aligned}$$
ce qui constitue la conclusion recherchée.

3. Enfin, nous avons :
$$\lim_{\sigma\to 0}\widehat{h}_\sigma(\nu) = \lim_{\sigma\to 0}\mathrm{e}^{2\pi^2\sigma^2\nu^2} = 1.$$
Par convergence dominée, ceci donne :
$$\lim_{\sigma\to 0}\int_{\mathbb{R}}\widehat{s}(\nu)\widehat{h_\sigma}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu = \int_{\mathbb{R}}\widehat{s}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu.$$
Il reste donc à montrer :
$$\lim_{\sigma\to 0} s\star h_\sigma = s,$$
dans L^1. Pour ce faire, notons que :
$$\int_{\mathbb{R}}|s\star h_\sigma - s| = \int_{\mathbb{R}}|\int_{\mathbb{R}}\big(s(t-u)-s(t)\big)h_\sigma(u)\mathrm{d}u|\mathrm{d}t.$$
Donc, en posant $\phi(u) = \int_{\mathbb{R}}|s(t-u)-s(t)|\mathrm{d}t$, nous obtenons :
$$\int_{\mathbb{R}}|s\star h_\sigma - s| \leq \int_{\mathbb{R}}\phi(u)h_\sigma(u)\mathrm{d}u = \int_{\mathbb{R}}\phi(\sigma u)h_1(u)\mathrm{d}u = I_\sigma.$$
On conclut avec le raisonnement par densité suivant :

– Si s est à support compact et continue, alors par uniforme-continuité, pour toute suite u_n tendant vers 0

$$\lim_{n\to+\infty} |s(t-u_n)-s(t)| = 0.$$

Donc, puisque ϕ est intégrable, on obtient par convergence dominée :

$$\lim_{\sigma\to 0} I_\sigma = 0.$$

– Si s est seulement supposé stable, alors il existe une suite de fonctions s_n à support compact convergent vers s dans L^1. On obtient alors le résultat cherché par l'intermédiaire de cette approximation.

□

1.2 Séries de Fourier des signaux L^1_{loc}

On s'intéresse maintenant aux séries de Fourier. Celles-ci peuvent être vues comme des transformées de Fourier particulières : il s'agit en effet de transformées de Fourier de fonctions périodiques, qui par conséquent ne sont pas L^1. Dans la suite, L^1_{loc} désigne l'espace des signaux localement stable, i.e. :

$$L^1_{loc} := \left\{ s / \forall A \text{ compact} \subset \mathbb{R},\ \mathbb{1}_A . s \in L^1 \right\}.$$

1.2.1 Définitions, coefficients de Fourier

On rappelle le cadre fonctionnel considéré.

Définition 3. *Soit $T > 0$. Un signal s est dit T-périodique si :*

$$\forall t \in \mathbb{R}, s(t+T) = s(t).$$

Un signal s T-périodique est dit de plus localement stable si $s \in L^1_{loc}$.

Et dans le cas des fonctions périodiques, la transformée de Fourier est définie sur un ensemble discret, c'est la suite des coefficients de Fourier.

Définition 4. *La transformée de Fourier $(\widehat{s}_n)_{n\in\mathbb{Z}}$ d'un signal localement stable T-périodique s est définie par la formule :*

$$\widehat{s}_n := \frac{1}{T}\int_0^T s(t)\mathrm{e}^{-2\pi i \frac{n}{T}t}\mathrm{d}t.$$

Étant donné $n \in \mathbb{Z}$, $\widehat{s}_n$ est appelé n-ième coefficient de Fourier du signal s.

1.2.2 Convolution

Puisque le produit de convolution est une opération essentiellement algébrique, elle s'applique au cadre précédemment défini. On peut même convoler des fonctions périodiques et des fonctions L^1, comme l'explique le résultat suivant.

Lemme 3. *Soit s un signal localement stable T-périodique et h un signal stable. Alors la fonction g définie par la formule :*

$$g(t) := \int_{\mathbb{R}} h(t-u)s(u)\mathrm{d}u$$

est T-périodique, définie presque partout et localement stable.
De plus son n-ième coefficient de Fourier est donné par la formule :

$$\widehat{g}_n = \widehat{h}(\frac{n}{T})\widehat{s}_n.$$

Cette dernière formule est à mettre en relation avec le fait que la transformée de Fourier change un produit de convolution en produit.

Démonstration. Par changement de variable, on a :

$$\int_{\mathbb{R}} h(t-u)s(u)\mathrm{d}u = \int_0^T \widetilde{h}_T(t-u)s(u)\mathrm{d}u,$$

où l'on a noté $\widetilde{h}_T(u) = \sum_{n\in\mathbb{Z}} h(u+nT)$. Comme :

$$\int_0^T |\widetilde{h}_T(u)|\mathrm{d}u \leq \int_{\mathbb{R}} |h(u)|\mathrm{d}u < +\infty,$$

on a $\int_{\mathbb{R}} |\widetilde{h}_T(t-u)||s(u)|\mathrm{d}u < +\infty$ presque partout, et g est définie presque partout. De plus :

$$g(t+T) = \int_{\mathbb{R}} h(t+T-u)s(u)\mathrm{d}u = \int_{\mathbb{R}} h(t-u)s(u)\mathrm{d}u,$$

et donc g est T-périodique. On vérifie aisément que g est localement stable. Enfin :

$$\begin{aligned}
\widehat{g}_n &= \frac{1}{T}\int_0^T g(t)\mathrm{e}^{2i\pi\frac{n}{T}t}\mathrm{d}t \\
&= \frac{1}{T}\int_0^T \int_0^T \widetilde{h}_T(t-u)s(u)\mathrm{e}^{2i\pi\frac{n}{T}t}\mathrm{d}u\mathrm{d}t \\
&= \frac{1}{T}\int_0^T \int_0^T \left(\widetilde{h}_T(t-u)\mathrm{e}^{2i\pi\frac{n}{T}(t-u)}\right) s(u)\mathrm{e}^{-2i\pi\frac{n}{T}u}\mathrm{d}u\mathrm{d}t \\
&= \widehat{h}(\frac{n}{T})\widehat{s}_n.
\end{aligned}$$

□

La formule que nous venons d'obtenir sur les coefficients peut sembler sophistiquée, mais elle nous sera utile par la suite.

1.2.3 Le Noyau de Poisson et son application à la formule d'inversion

Dans cette section nous établissons la formule de Poisson, qui est une des briques élémentaires de la preuve du Théorème de Shannon[5]-Nyquist[6] qui constitue notre but ultime pour cette partie.

Noyau de Poisson

On introduit tout d'abord le noyau de Poisson et quelques-unes de ses propriétés.

Définition 5. *Étant donné un réel r appartenant à $]-1,1[$ et $T>0$, on appelle noyau de Poisson la fonction définie sur $\mathbb{R}$ et à valeur dans $\mathbb{C}$ définie par la formule :*

$$P_r(t) := \sum_{n\in\mathbb{Z}} r^{|n|}\mathrm{e}^{2i\pi\frac{n}{T}t}.$$

Certaines propriétés de ce noyau sont indiquées dans la proposition suivante.

Proposition 1. *La fonction P_r vérifie :*

1. $P_r(t) = \sum_{n\in\mathbb{Z}} r^{|n|}\mathrm{e}^{2i\pi\frac{n}{T}t} + \sum_{n\in\mathbb{Z}} r^{|n|}\mathrm{e}^{-2i\pi\frac{n}{T}t} - 1 = \frac{1-r^2}{|1-r\mathrm{e}^{2i\pi\frac{t}{T}}|^2}$.

[5]Claude Elwood Shannon (30 avril 1916 Gaylord, Michigan - 24 février 2001) est un ingénieur électricien et mathématicien américain. Il est l'un des pères, si ce n'est le père fondateur, de la théorie de l'information.

[6]Harry Nyquist (7 février 1889 Nilsby, Suède- 4 avril 1976, Harlingen, Texas) a été un important contributeur à la théorie de l'information et à l'automatique. Ses travaux théoriques sur la détermination de la bande passante nécessaire à la transmission d'information, publiés dans l'article "Certain factors affecting telegraph speed" posent les bases des recherches de Claude Shannon.

2. *Pour tout réel t, on a :*
$$P_r(t) \geq 0.$$
3. $\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} P_r(t)\mathrm{d}t = 1.$
4. *Pour tout $\varepsilon > 0$, on a :*
$$\frac{1}{T}\int_{[-T/2,T/2]-[-\varepsilon,\varepsilon]} P_r(t)\mathrm{d}t \leq \frac{1-r^2}{|1-r\mathrm{e}^{2i\pi\frac{\varepsilon}{T}}|^2} \xrightarrow[r\to 1]{} 0.$$

Les propriétés 2, 3, 4 correspondent à ce que l'on appelle une *identité approchée*. On a déjà rencontré cette notion, sans le signaler, avec la fonction h_σ de la preuve du théorème 1. Les identités approchées permettent de régulariser les fonctions par convolution et d'utiliser ensuite des formules du type :
$$\lim_{r\to 1}\frac{1}{T}\int_0^T \varphi(t)P_r(t)\mathrm{d}t = \varphi(0), \tag{1.1}$$
lorsque l'on souhaite revenir à la fonction initiale. C'est ce raisonnement que nous allons maintenant effectuer avec le noyau de Poisson pour obtenir une formule d'inversion.

Théorème 2. *Soit s un signal T-périodique localement stable, tel que*
$$\sum_{n\in\mathbb{Z}} |\widehat{s}_n| < +\infty.$$
Alors, pour presque tout $t \in \mathbb{R}$,
$$s(t) = \sum_{n\in\mathbb{Z}} \widehat{s}_n \mathrm{e}^{2i\pi\frac{n}{T}t}.$$

Autrement dit, si $\sum_{n\in\mathbb{Z}} |\widehat{s}_n| < +\infty$, la fonction s est connue dès lors que ses coefficients de Fourier sont connus.

Remarque 1. *Si on suppose en outre que s est continue, alors la formule d'inversion devient vraie pour tout $t \in \mathbb{R}$.*

Démonstration. On commence par appliquer le résultat du Lemme 3 :
$$\sum_{n\in\mathbb{Z}} \widehat{s}_n r^{-|n|}\mathrm{e}^{2i\pi\frac{n}{T}t} = \frac{1}{T}\int_0^T s(u)P_r(t-u)\mathrm{d}u,$$
puisque toutes les intégrales considérées sont convergentes. De plus :
$$\lim_{r\to 1}\int_0^T |\frac{1}{T}\int_0^T s(u)P_r(t-u)\mathrm{d}u - s(t)|\mathrm{d}t = 0,$$
d'après (1.1) et le théorème de convergence dominée. Par conséquent la suite de fonctions $\varphi_r(t) := \sum_{n\in\mathbb{Z}} \widehat{s}_n r^{-|n|}\mathrm{e}^{2i\pi\frac{n}{T}t}$ (indexée par r) converge dans L^1_{loc} vers s. D'autre part, puisque $\sum_{n\in\mathbb{Z}} |\widehat{s}_n| < +\infty$, la fonction $\varphi_r(t)$ tend vers $\sum_{n\in\mathbb{Z}} \widehat{s}_n \mathrm{e}^{2i\pi\frac{n}{T}t}$ ponctuellement. On utilise alors le résultat du cours d'intégration suivant :

"Si une suite de fonction f_n converge vers f dans L^p et vers g presque partout, alors $f = g$ presque partout"

pour conclure que pour presque tout $t \in \mathbb{R}$:
$$s(t) = \sum_{n\in\mathbb{Z}} \widehat{s}_n \mathrm{e}^{2i\pi\frac{n}{T}t}.$$

□

Au passage, on en déduit le résultat suivant.

Corollaire 1. *2 signaux T-périodiques stable ayant les mêmes coefficients de Fourier sont égaux presque partout.*

Formule de Poisson faible

La formule que nous allons établir est une version faible du résultat suivant :
Pour tout $s \in L^1$,

$$T\sum_{n\in\mathbb{Z}} s(nT) = \sum_{n\in\mathbb{Z}} \widehat{s}(\frac{n}{T}).$$

Théorème 3. *Soit s un signal stable et $T > 0$. La série $\sum_{n\in\mathbb{Z}} s(t+nT)$ converge presque partout vers une fonction ϕ T-périodique, localement intégrable et dont le n-ième coefficient de Fourier est donné par la formule :*

$$\widehat{\phi}_n = \frac{1}{T}\widehat{s}(\frac{n}{T}).$$

Ce théorème établit donc un lien entre séries de Fourier et transformée de Fourier. Formellement, il nous dit que ϕ est T-périodique et que sa série de Fourier formelle s_f est :

$$s_f(t) = \frac{1}{T}\sum_{n\in\mathbb{Z}} \widehat{s}(\frac{n}{T})\mathrm{e}^{2i\frac{\pi}{T}t}.$$

Remarque 2. *Si on arrive à montrer $\phi(0) = s_f(0)$, alors on aura la formule de Poisson.*

Ce résultat faible nous suffira pour établir le théorème de Shannon-Nyquist dans la section suivante.

Démonstration. Notons tout d'abord que ϕ est bien définie, puisque :

$$\begin{aligned}\int_0^T \sum_{n\in\mathbb{Z}} |s(t+nT)|\mathrm{d}t &= \sum_{n\in\mathbb{Z}} \int_0^T |s(t+nT)|\mathrm{d}t \\ &= \sum_{n\in\mathbb{Z}} \int_{nT}^{(n+1)T} |s(t)|\mathrm{d}t \\ &= \int_{\mathbb{R}} |s(t)|\mathrm{d}t < +\infty, \end{aligned} \tag{1.2}$$

et que donc $\sum_{n\in\mathbb{Z}} |s(t+nT)| < +\infty$ presque partout.
On montre aisément que ϕ est T-périodique. La formule (1.2) montre en outre que ϕ est localement intégrable. On peut donc calculer son coefficient de Fourier. On obtient :

$$\begin{aligned}\widehat{\phi}_n &= \frac{1}{T}\int_0^T \phi(t)\mathrm{e}^{-2i\pi\frac{n}{T}t}\mathrm{d}t \\ &= \frac{1}{T}\int_0^T \left(\sum_{n\in\mathbb{Z}} s(t+nT)\right)\mathrm{e}^{-2i\pi\frac{n}{T}t}\mathrm{d}t \\ &= \frac{1}{T}\int_0^T \left(\sum_{n\in\mathbb{Z}} s(t+nT)\mathrm{e}^{-2i\pi\frac{n}{T}(t+nT)}\right)\mathrm{d}t \\ &= \frac{1}{T}\int_0^T s(t)\mathrm{e}^{-2i\pi\frac{n}{T}(t)}\mathrm{d}t \\ &= \frac{1}{T}\widehat{s}(\frac{n}{T}).\end{aligned}$$

□

Pour aller un peu plus loin vers la version forte, on peut énoncer le résultat suivant.

Proposition 2. *Soit s un signal stable tel que $\sum_{n\in\mathbb{Z}} |\widehat{s}(\frac{n}{T})| < +\infty$. Alors :*

$$\sum_{n\in\mathbb{Z}} s(t+nT) = \frac{1}{T}\sum_{n\in\mathbb{Z}} \widehat{s}(\frac{n}{T})\mathrm{e}^{-2i\pi\frac{n}{T}t}.$$

Ce résultat est en fait une simple application de la formule d'inversion obtenue au théorème 2.

1.3 Reconstruction des signaux périodiques

Dans cette dernière section nous allons répondre aux deux questions suivantes :

1. Étant donné un signal continu, quels critères doivent-être appliqués pour en extraire une suite discrète représentative ?
2. Comment reconstruire un signal continu à partir d'un échantillon discret de valeurs ?

Les réponses à ces deux questions sont en partie données par le théorème de Shannon-Nyquist, lui-même obtenu à partir de la formule de Poisson vue à la section précédente.

1.3.1 Le théorème de Shannon-Nyquist

Avant d'énoncer le théorème, on montre deux résultats préliminaires.

Lemme 4. *Soit s, un signal stable et continu, de transformée de Fourier $\widehat{s}$ stable et un réel $B > 0$. Supposons que :*

$$\sum_{n\in\mathbb{Z}} |s(\frac{n}{2B})| < +\infty. \tag{1.3}$$

Alors :

$$\sum_{j\in\mathbb{Z}} \widehat{s}(\nu + 2jB) = \frac{1}{2B}\sum_{n\in\mathbb{Z}} s(\frac{n}{2B})\mathrm{e}^{-2i\pi\nu\frac{n}{2B}},$$

pour presque tout $\nu \in \mathbb{R}$.

Démonstration. D'après la formule de Poisson faible, la fonction $\phi(\nu) = \sum_{j\in\mathbb{Z}} \widehat{s}(\nu + 2jB)$ est localement intégrable et son n-ième coefficient de Fourier vaut :

$$\widehat{\phi}_n = \frac{1}{2B}\int_{\mathbb{R}} \widehat{s}(\nu)\mathrm{e}^{-2i\pi\frac{n\nu}{2B}}\mathrm{d}\nu.$$

Mais puisque $\widehat{s}$ est stable, on peut appliquer la formule d'inversion du théorème 1. Dans notre cas, celle-ci s'écrit :

$$\frac{1}{2B}\int_{\mathbb{R}} \widehat{s}(\frac{n}{2B})\mathrm{e}^{-2i\pi\frac{n\nu}{2B}}\mathrm{d}\nu = s(-\frac{n}{2B}).$$

En combinant les deux formules, nous obtenons donc formellement :

$$\phi(\nu) = \frac{1}{2B}\sum_{n\in\mathbb{Z}} s(\frac{n}{2B})\mathrm{e}^{-2i\pi\frac{n\nu}{2B}}.$$

Mais l'hypothèse (1.3), permet l'utilisation du théorème d'inversion 2 et donc justifie rigoureusement cette dernière formule. □

Le lemme suivant donne un résultat technique.

Lemme 5. *Soit s un signal vérifiant les hypothèses du lemme 4 et h un signal de la forme :*

$$h(t) = \int_{\mathbb{R}} T(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu, \tag{1.4}$$

où T est stable. Alors, le signal :

$$\widetilde{s}(t) = \frac{1}{2B}\sum_{n\in\mathbb{Z}} s(\frac{n}{2B})h(t - \frac{n}{2B}) \tag{1.5}$$

admet la représentation :

$$\widetilde{s}(t) = \int_{\mathbb{R}} \left(\sum_{n\in\mathbb{Z}} \widehat{s}(\nu + 2jB)\right) T(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu.$$

Démonstration. Notons tout d'abord que $\widetilde{s}$ est borné et continu, en tant que somme d'une série normalement convergente.
En remplaçant h par sa valeur dans (1.5), on obtient :

$$\begin{aligned}\widetilde{s}(t) &= \frac{1}{2B}\sum_{n\in\mathbb{Z}} s(\frac{n}{2B})\int_{\mathbb{R}} T(\nu)\mathrm{e}^{2i\pi\nu(t-\frac{n}{2B})}\mathrm{d}\nu \\ &= \int_{\mathbb{R}}\sum_{n\in\mathbb{Z}}\frac{1}{2B}s(\frac{n}{2B})\mathrm{e}^{-\frac{2i\pi\nu n}{2B}}T(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu.\end{aligned}$$

On utilisé la théorème de Fubini, ce qui est justifié, car :

$$\int_{\mathbb{R}}\sum_{n\in\mathbb{Z}}|s(\frac{n}{2B})|.|T(\nu)|\mathrm{d}\nu = \left(\sum_{n\in\mathbb{Z}}|s(\frac{n}{2B})|\right).\left(\int_{\mathbb{R}}|T(\nu)|\mathrm{d}\nu\right) < +\infty.$$

Donc :

$$\widetilde{s}(t) = \int_{\mathbb{R}} g(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu,$$

avec :

$$g(\nu) = \sum_{n\in\mathbb{Z}}\frac{1}{2B}s(\frac{n}{2B})\mathrm{e}^{-\frac{2i\pi\nu n}{2B}}T(\nu),$$

ou encore, d'après le théorème précédent :

$$g(\nu) = \sum_{j\in\mathbb{Z}}\widehat{s}(\nu+2jB),$$

ce qui conduit au résultat annoncé. □

On peut alors énoncer le fameux théorème de Shannon-Nyquist.

Théorème 4. *(de Shannon-Nyquist) Soit s, un signal stable et continu dont la transformée de Fourier $\widehat{s}$ s'annule en dehors d'un intervalle $[-B,B]$. Supposons également que* (1.3) *soit vérifiée. Alors,*

$$s(t) = \sum_{n\in\mathbb{Z}} s(\frac{n}{2B})\mathrm{sinc}\Big((2Bt-n)\pi\Big). \tag{1.6}$$

Question : quel est l'intérêt immédiat de cette formule ?

Démonstration. Dans le théorème précédent, spécifions T par la formule :

$$T := \mathbb{1}_{[-B,B]},$$

de telle sorte que :

$$h(t) = 2B\mathrm{sinc}(2B\pi\nu) = 2B\frac{\sin(\pi\nu 2B)}{\pi\nu 2B}.$$

Alors :

$$\left(\sum_{j\in\mathbb{Z}}\widehat{s}(\nu+2jB)\right)T(\nu) = \widehat{s}(\nu)T(\nu) = \widehat{s}(\nu),$$

d'après le choix de T et l'hypothèse faite sur le support de $\widehat{s}$. On applique alors le théorème précédent, ce qui donne :

$$\widetilde{s}(t) = \int_{\mathbb{R}}\widehat{s}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu = \frac{1}{2B}\sum_{n\in\mathbb{Z}} s(\frac{n}{2B})h(t-\frac{n}{2B}),$$

puis, par le théorème d'inversion 1, on obtient :

$$\int_{\mathbb{R}}\widehat{s}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu = s(t),$$

ce qui achève la démonstration. □

La fonction T joue donc finalement le rôle d'une troncature, puisqu'elle tronque artificiellement le support de la fonction considérée.

1.3.2 Phénomène de recouvrement de spectre

Le théorème de Shannon-Nyquist nous indique comment choisir une fréquence d'échantillonnage lorsqu'on a des informations sur le support de la transformée de Fourier du signal[7] considérée. Si le support en question est inclus dans l'intervalle $[-B, B]$, alors, une fréquence d'échantillonnage correcte est $f_e = \frac{1}{2B}$, parfois appelée *fréquence de Nyquist*. Mais que se passe-t-il lorsqu'on sous-échantillonne, c'est-à-dire, lorsque l'hypothèse faite sur le support n'est plus valide ?
Supposons donc que l'affirmation $\mathrm{supp}(\widehat{s}) \subset [-B, B]$ soit fausse. Après multiplication de $\widehat{s}$ par $\mathbb{1}_{[-B,B]}$, c'est-à-dire convolution par $h(t) = 2B\mathrm{sinc}(2\pi Bt)$, on se ramène au cadre précédent et on a :

$$\widetilde{s}(t) = \sum_{n\in\mathbb{Z}} s(\frac{n}{2B})\mathrm{sinc}\big((2Bt - n)\pi\big).$$

Quelle est la transformée de Fourier de ce signal ?

Proposition 3. *Soit s un signal stable et continu tel que la condition* (1.3) *soit satisfaite. Alors le signal :*

$$\widetilde{s}(t) = \sum_{n\in\mathbb{Z}} s(\frac{n}{2B})\mathrm{sinc}\big((2BT - n)\pi\big),$$

admet la représentation :

$$\widetilde{s}(t) = \int_{\mathbb{R}} \widehat{\widetilde{s}}(\nu)\mathrm{e}^{2i\pi\nu t}\mathrm{d}\nu,$$

avec :

$$\widehat{\widetilde{s}}(\nu) = \left(\sum_{j\in\mathbb{Z}} \widehat{s}(\nu + 2jB)\right) \mathbb{1}_{[-B,B]}(\nu). \tag{1.7}$$

La démonstration de ce résultat s'obtient en reprenant ligne à ligne celle du théorème 4.
En conséquence, on voit que les supports des fonctions dans la somme (1.7) vont se recouvrir sur les bords de l'intervalle $[-B, B]$, ce qui altérera le spectre du signal reconstruit et par conséquent, le signal lui-même. Ce phénomène est appelé *recouvrement du spectre* ou encore, en anglais, *aliasing*.
La conclusion pratique que l'on peut tirer de cette proposition est que, lorsque l'on souhaite discrétiser[8] un signal, il faut tout d'abord en tronquer le spectre- à l'aide d'un filtre, voir les chapitres 5 et 6 - puis l'échantillonner à une fréquence au moins égale à celle de Nyquist.
Un exemple simple est celui de la numérisation de signaux sonores. Les fréquences audibles pour un nourrisson vont de 20Hz à 20kHz. Pour concevoir le format de codage des CD audio, les ingénieurs ont choisi une fréquence d'échantillonnage de 44kHz, c'est-à-dire à peu près le double de la fréquence maximale audible, ce qui correspond bien à la fréquence de Nyquist. Les 4kHz supplémentaires correspondent à des corrections d'erreurs qui seront présentées au chapitre 3. De même, l'échantillonnage des conversations téléphoniques se fait à la cadence de 8kHz, qui est plus faible que celle des CD audio, car les spectres considérés sont beaucoup plus étroits et la qualité du rendu n'est plus un critère prédominant.

1.3.3 Sur-échantillonnage

On peut maintenant se poser la question inverse. Quelles sont les conséquences d'un échantillonnage à une fréquence trop élevée? Avant de répondre à cette question, remarquons que dans la formule (1.6) la série obtenu converge très lentement, grosso-modo en $\frac{1}{n}$. Elle n'est donc pas satisfaisante d'un point de vue pratique.
Supposons que l'on sur-échantillonne un signal s, c'est-à-dire que son spectre vérifie $\mathrm{supp}(\widehat{s}) \subset [-W, W]$, avec $B = (1 + \alpha)W$, pour un certain $\alpha > 0$. Choisissons notre fonction de troncature T de telle sorte que :

$$\forall \nu \in [-W, W], \qquad T(\nu) = 1,$$

[7] Les ingénieurs parlent plutôt de spectre d'un signal.
[8] En utilisant un vocabulaire plus proche de celui des ingénieurs, on dirait "numériser".

et

$$\forall \nu \in \mathbb{R} - [-B, B], \qquad T(\nu) = 0,$$

alors le théorème de Shannon-Nyquist s'applique et en en reproduisant la preuve, on obtient :

$$s(t) = \frac{1}{2B} \sum_{n \in \mathbb{Z}} s(\frac{n}{2B}) h(t - \frac{n}{2B}),$$

où la fonction h est la transformée inverse de T donnée par la formule (1.4). On met ainsi en évidence un nouveau degré de liberté : par un choix judicieux de h, i.e. de T, on peut accélérer la vitesse de convergence. Ceci se fait en particulier en augmentant la régularité de la fonction T. Une première amélioration est par exemple obtenue dans notre cas en choisissant :

$$T(\nu) = \frac{B + \nu}{B - W} \mathbb{1}_{[-B, -W]}(\nu) + \mathbb{1}_{]-W, W[}(\nu) + \frac{B - \nu}{B - W} \mathbb{1}_{[W, B]}(\nu).$$

1.4 Note bibliographique

Pour plus détails sur ce chapitre, on consultera les parties A et B de la référence [1].

Chapitre 2

Quantification scalaire des signaux discrets

2.1 Introduction

Après avoir échantillonné un signal analogique, on dispose d'un signal discret, c'est-à-dire une suite de nombres réels. A ce stade, la conversion analogique-numérique (souvent désignée par *CAN*) n'est pas encore achevée. Il faut expliquer comment on transforme une suite de nombres réels en une suite de "0" et de "1". C'est l'objet de ce chapitre.
On considère donc une *source*[1] émettant un signal discret en temps. La transformation des nombres émis en suite de nombres binaires consiste en fait en deux opérations : l'*encodage* et le *décodage*.
Encoder un signal, c'est appliquer à chacun des termes de la suite qu'il constitue une fonction de la forme :

$$\begin{array}{rcl} Q : [-M, M] & \rightarrow & \{I_n\}_{0 \leq n \leq N} \\ s(t) & \rightarrow & I_n, \end{array}$$

où :

- l'ensemble $\{I_n\}_{0 \leq n \leq N}$ est une famille d'intervalles disjoints formant une partition de l'intervalle $[-M, M]$ (N est un entier naturel),
- $s(t)$ est un réel, représentant la valeur du signal à l'instant t, supposé appartenir à l'intervalle $[-M, M]$ (M est un réel).

L'encodage est donc une opération irréversible car faisant intervenir une fonction non-injective, qui entraîne une perte d'information.
Un exemple courant est la quantification sur 3 bits. Dans ce cas, l'intervalle $[-M, M]$ est partitionné en $N = 2^3 = 8$ intervalles.
Décoder un signal, c'est réaliser l'opération inverse, c'est-à-dire appliquer à chacun des termes d'une suite d'intervalles une fonction de la forme :

$$I_n \rightarrow y_n,$$

où y_n est un réel représentant l'intervalle I_n, par exemple la valeur de son milieu.

Remarque 3. *Les valeurs y_n sont ensuite elles-même codées par des entiers binaires.*

La figure 2.1 représente un signal avant et après quantification.

2.2 Formulation du problème

Dans cette section on formalise mathématiquement le problème de la quantification.

[1] Ce terme sera très largement employé dans la suite et désigne un dispositif émettant des signaux à partir de maintenant supposés discrets en temps.

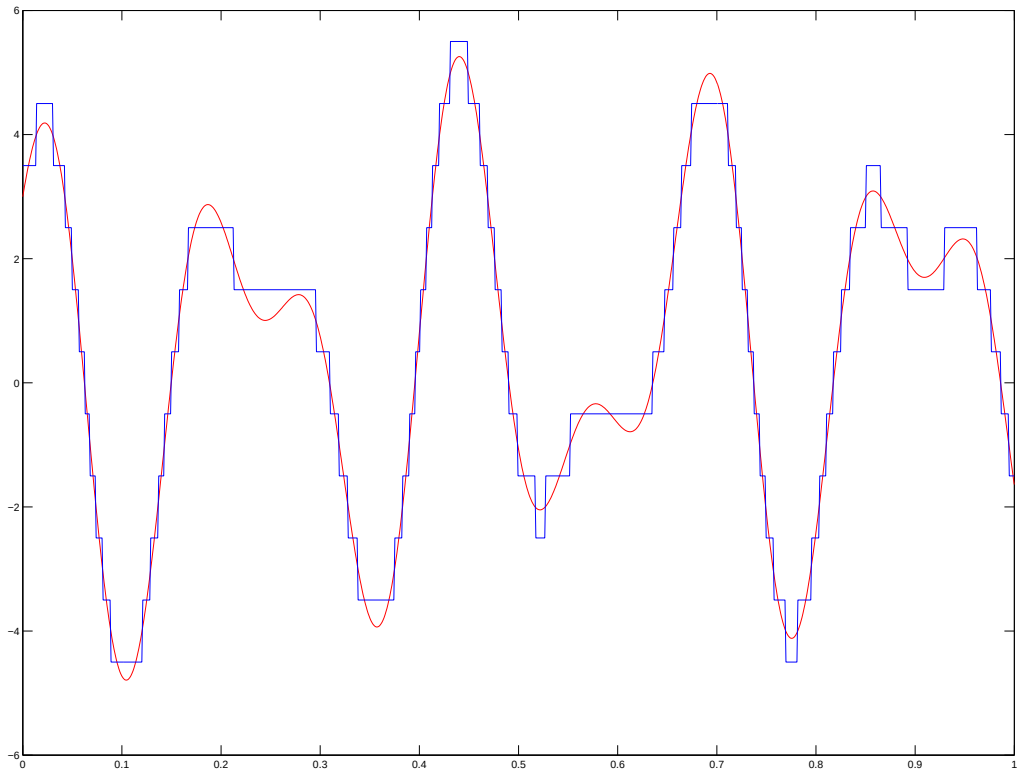

FIG. 2.1 – Exemple de quantification d'un signal.

2.2.1 Cadre

On considère donc une source émettant un signal aléatoire S de densité de probabilité $f_S(x)$. Pour construire une quantification, il faut donc définir une famille d'intervalles (i.e. la famille $\{I_n\}_{0\leq n\leq N}$ de l'introduction), donc de frontières, et de valeurs d'assignation (i.e. les valeurs y_n de l'introduction). Pour fixer les idées conservons les notations de l'introduction et désignons par N le nombre d'intervalles constituant la partition et donc également le nombre de valeurs d'assignations. Le nombre de valeurs frontières, aussi appelées *valeurs de décision*, que l'on note b_i est alors $N+1$. On pose a priori $b_0 = -\infty$ et $b_N = +\infty$ et on signalera dans la suite les cas où l'on adopte une autre convention. La fonction de quantification s'écrit :

$$Q(x) = y_n,$$

pour $x \in]b_{n-1}, b_n]$.

2.2.2 Erreur de distorsion de quantification

On définit alors l'erreur quadratique moyenne de quantification par :

$$\begin{aligned}\sigma_q^2(B,Y) &= \int_{-\infty}^{+\infty}(x-Q(x))^2 f_S(x)\mathrm{d}x \\ &= \sum_{n=1}^{N}\int_{b_{n-1}}^{b_n}(x-y_n)^2 f_S(x)\mathrm{d}x,\end{aligned}$$

où l'on a noté $B = \{b_n\}_{0 \leq n \leq N}$ et $Y = \{y_n\}_{1 \leq n \leq N}$. La quantité σ_q est aussi appelée *erreur de distorsion de quantification*. Une formalisation possible de l'erreur de quantification consiste à considérer son effet comme celui d'un bruit externe affectant le signal S.
Le problème est alors le suivant :
Étant donné f_S et N le nombre d'intervalles de quantification, trouver les valeurs b_n et y_n solution du problème

$$\min_{B,Y} \sigma_q(B, Y).$$

En conclusion, l'erreur de distorsion de quantification dépend à la fois de la partition choisie et du choix du représentant.

2.2.3 Codage en taille variable

Si on considère un codage des y_n en entiers binaires de taille variable ℓ_n, la taille moyenne des symboles après codage est :

$$R = \sum_{n=1}^{N} \ell_n p(y_n), \tag{2.1}$$

où l'on a noté $p(y_n)$ la probabilité d'apparition après codage du symbole correspondant à y_n. Attention, la valeur R dépend des frontières car :

$$p(y_n) = \int_{b_{n-1}}^{b_n} f_S(x)\mathrm{d}x,$$

et donc :

$$R = \sum_{n=1}^{N} \ell_n \int_{b_{n-1}}^{b_n} f_S(x)\mathrm{d}x.$$

On considère donc selon les cas l'une ou l'autre des reformulations du problème suivantes.

1. Si l'on se donne une contrainte de distorsion de quantification

$$\sigma_q \leq \sigma^\star, \tag{2.2}$$

   où $\sigma^\star$ est un réel fixé, trouver le nombre N, les frontières B et les représentants Y minimisant le taux R, défini par (2.1) et satisfaisant (2.2).

2. Si l'on se donne une contrainte sur le taux

$$R \leq R^\star, \tag{2.3}$$

   où $R^\star$ est un réel fixé, trouver le nombre N, les frontières B et les représentants Y minimisant la distorsion quantification et satisfaisant (2.3).

L'une où l'autre de ces formulations constitue un problème plus général que celui que nous allons considérer dans ce chapitre. On se restreint en effet dans la suite à de codage de tailles fixes, si bien que R est constant par rapport à B et Y. Le problème du codage en taille variable sera quant à lui traité au chapitre 3.

2.3 Quantification uniforme

La solution de quantification la plus simple est la quantification uniforme. Dans ce cadre, tout les intervalles ont la même longueur, que l'on note dans la suite Δ. On parle dans ce cas de *quantification uniforme*. On suppose que l'on adopte une stratégie de codage où N le nombre d'intervalles est pair, c'est-à-dire que $0 \in B$.

2.3.1 Quantification uniforme des sources uniformes

Supposons que pour tout t, $s(t) \in [-S_{max}, S_{max}]$ avec une probabilité uniforme. Dans ce cas, on fixe cette fois-ci $b_0 = -S_{max}$ et $b_N = S_{max}$. Alors $\Delta = \frac{2S_{max}}{N}$ et

$$\begin{aligned}\sigma_q^2(B,Y) &= 2\sum_{n=1}^{N/2}\int_{(n-1)\Delta}^{n\Delta}(x-\frac{2n-1}{2}\Delta)^2\frac{1}{2S_{max}}\mathrm{d}x \\ &= \frac{\Delta^2}{12}.\end{aligned}$$

Afin d'évaluer l'effet de l'augmentation du nombre d'intervalle sur la qualité du codage, on considère alors le rapport signal/bruit défini par :

$$SNR = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right),$$

où σ_x désigne l'écart type du signal S.
Puisque S est une variable aléatoire de loi uniforme à valeurs dans $[-S_{max}, S_{max}]$, on a alors :

$$\sigma_x^2 = \frac{1}{2S_{max}}\int_{-S_{max}}^{S_{max}} x^2\mathrm{d}x = \frac{2S_{max}^3}{12} = \frac{S_{max}^2}{3}.$$

Par conséquent, le rapport signal/bruit vaut :

$$\begin{aligned}SNR &= 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) \\ &= 10\log_{10}\left(\frac{(2S_{max})^2}{12}.\frac{12}{(\frac{2S_{max}}{N})^2}\right) \\ &= 10\log_{10}(N^2) \\ &= 20\log_{10}(2^k) \\ &= 6,02kdB,\end{aligned}$$

où k représente le nombre de bits utilisés pour le codage binaire des représentants y_n.
La conclusion du calcul précédent est que l'ajout d'un bit de quantification augmente le rapport signal/bruit d'approximativement $6dB$.

2.3.2 Quantification uniforme des sources non-uniformes

Pour introduire ce qui va suivre, considérons l'exemple d'une source émettant dans $[-100, 100]$ dont 95% des valeurs sont dans $[-1, 1]$. Supposons que l'on ait adopté la stratégie de quantification uniforme de la section précédente et que le codage soit effectué sur $k = 3$bits, ce qui revient à considérer 8 intervalles de longueur 25. On a donc, dans 95% des cas une erreur minimum de 11,5. Cet exemple est représentatif des sources gaussiennes et montre le défaut de la quantification uniforme telle que présentée dans la section précédente.
Si l'on souhaite rester dans le cadre de la quantification uniforme, une solution consiste à utiliser la fonction de répartition pour optimiser la taille des intervalles considérés. Concrètement cela revient à considérer σ_q^2 comme une fonction de Δ et à résoudre :

$$\min_{\Delta}\sigma_q^2(\Delta).$$

Explicitons le début du calcul menant à la résolution de ce problème. On a :

$$\begin{aligned}\sigma_q^2(\Delta) &= 2\sum_{n=1}^{\frac{N}{2}-1}\int_{(n-1)\Delta}^{n\Delta}\left(x-\frac{2n-1}{2}\Delta\right)^2 f_S(x)\mathrm{d}x \\ &\quad + 2\int_{(\frac{N}{2}-1)\Delta}^{+\infty}\left(x-\frac{2n-1}{2}\Delta\right)^2 f_S(x)\mathrm{d}x.\end{aligned}$$

On peut alors calculer la dérivée de cette fonction :

$$\begin{aligned}\frac{\partial\sigma_q^2(\Delta)}{\partial\Delta} &= -\sum_{n=1}^{\frac{N}{2}-1}(2n-1)\int_{(n-1)\Delta}^{n\Delta}\left(x-\frac{2n-1}{2}\Delta\right)f_S(x)\mathrm{d}x \\ &\quad +(N-1)\int_{(\frac{N}{2}-1)\Delta}^{+\infty}\left(x-\frac{2n-1}{2}\Delta\right)f_S(x)\mathrm{d}x.\end{aligned}$$

Pour trouver un extremum de σ_q, il faut donc annuler cette dernière valeur. L'équation en découlant n'est pas résoluble algébriquement dans le cas général. On la résout donc approximativement par des méthodes numériques. Il existe aussi des tableaux indiquant les solutions dans les cas de densités de probabilités classiques.

Remarque 4. *Dans ce cas, on peut en fait distinguer deux types d'erreurs de quantification :*

1. *L'erreur de surcharge, qui correspond aux erreurs se produisant dans les deux intervalles extrêmes. On parle également de bruit de saturation.*
2. *L'erreur granulaire, qui correspond à l'erreur de quantification dans les autres intervalles. On parle alors de bruit granulaire.*

Il arrive que l'on ne dispose que d'information partielles sur la source et sa densité de probabilité f_S. Cette carence conduit à des erreurs de modélisation. La fonction f_S^{approx} considérée ne coïncide pas avec la vraie fonction f_S. Il faut donc en fait prévoir une série de tests sur la source pour estimer ses paramètres.

2.4 Quantification adaptative

Une solution simple aux problèmes évoqués dans la section précédente consiste à fonder la stratégie de codage sur des intervalles de longueurs variables. On parle dans ce cas de *quantification adaptative.* Le but est bien sûr d'adapter les paramètres de quantification à la source considérée.

2.4.1 Approches "online" et "offline"

Deux approches sont généralement considérées, l'approche *online* conduisant à une adaptation de la quantification "rétrograde", c'est-à-dire effectuée en même temps que la quantification elle-même et une adaptation *offline*, où les paramètres de la quantification sont calculés de manière "directe", c'est-à-dire en fixant a priori les paramètres de la quantification.

2.4.2 Quantification adaptative directe

Dans cette approche le signal émis est divisé en blocs temporels et chaque bloc est analysé avant la quantification. Les paramètres de la quantification sont calculés en fonction de l'analyse. On a donc, dans ce cas, besoin d'adjoindre à chaque bloc codé un bloc d'information supplémentaire permettant d'indiquer au décodeur les paramètres de quantification qui ont finalement été retenus.
Deux problèmes apparaissent dans cette approche.

1. Un problème de synchronisation, car deux types de données sont transmis : le code du signal et les blocs d'informations.
2. Un problème de choix de la taille des bloc de codage considérés. Il faut alors trouver un compromis entre des bloc grands, qui seront alors plus grossièrement décrits par les paramètres de quantifications retenus, et des blocs petits, qui conduiront à de nombreux blocs d'information.

Donnons maintenant un exemple de procédure d'estimation des paramètres d'un bloc à quantifier.
Étant donné un bloc de taille M, un estime sa variance[2] au voisinage du temps n par

$$\widehat{\sigma}_q^2 = \frac{1}{M}\sum_{i=0}^{M-1} x_{i+n}.$$

[2]Il s'agit en fait de ce qu'on appelle en probabilité la variance empirique.

Remarque 5. *Attention, il faudra aussi quantifier cette valeur dans le bloc d'information, car tout doit être in fine sous forme binaire !*

On peut alors adopter un modèle gaussien et utiliser, pour la quantification du bloc, la stratégie uniforme indiquée à la section 2.3.2.

2.4.3 Quantification adaptative rétrograde

Dans cette second approche, l'adaptation se fait à la sortie du quantificateur. Elle ne nécessite pas de bloc d'information supplémentaire.
La méthode consiste à fixer le modèle, par exemple gaussien, et a adapter au cours du temps la valeur de Δ en observant les histogrammes issus de la quantification.

2.5 Quantification non-uniforme

La dernière solution envisagée habituellement consiste à postuler une densité de probabilité, à considérer ensuite σ_q comme une fonction des $2N+1$ variables réelles contenues dans les variables Y et B et finalement à optimiser σ_q globalement. Un calcul de différentielle conduit alors aux équations d'optimalité :

$$\begin{aligned} y_n &= \frac{\int_{b_{n-1}}^{b_n} x f_S(x)\mathrm{d}x}{\int_{b_{n-1}}^{b_n} f_S(x)\mathrm{d}x}, \quad n = 1, ..., N \\ b_n &= \frac{y_{n+1} + y_n}{2}, \quad n = 1, ..., N-1 \end{aligned}$$

avec $b_0 = -\infty$ et $b_N = +\infty$.
De même que ce qu'à la section 2.3.2, ce système n'est, dans le cas général, par résoluble algébriquement. Il faut donc mettre en oeuvre une méthode numérique pour le résoudre approximativement.

2.6 Note bibliographique

Pour plus de détails sur les différentes techniques de quantification des signaux, on consultera le chapitre 9 de la référence [2].

Chapitre 3

Codage sans perte de l'information

Nous continuons de suivre le cheminement du signal. Après avoir été échantillonné puis quantifié, il est maintenant représenté par une suite de 0 et de 1. Nous allons voir comment on peut préparer sa transmission dans de bonnes conditions. Plus précisément, deux raffinements très utiles vont être présentés dans ce chapitre : tout d'abord ce qui est parfois appelé le *codage source*, i.e. une méthode de compression du signal, comme le fait par exemple le programme zip, ensuite, ce qui est parfois appelé le *codage canal*, i.e. l'ajout judicieux de *bits de correction* qui vont permettre de corriger les erreurs éventuelles qui vont affecter le signal au cours de sa transmission.

3.1 Codage source et compression sans perte

On appelle donc codage source l'ensemble des techniques permettant de compresser avec ou sans perte d'information un signal numérique. Ceci revient en fait à coder de manière astucieuse les symboles émis par une source de manière à réduire le nombre total de bits utilisés.
Dans cette section on démontre le théorème fondamental du codage source, qui indique une limite théorique de compression sans perte et on présente un algorithme, dû à Huffman[1], permettant d'approcher arbitrairement cette limite.

3.1.1 Définitions

On commence par poser deux définitions.

Définition 6. *On appelle source de symboles m-aire tout dispositif émettant des mots construits par concaténations de symboles pris dans un alphabet de taille m.*

Par exemple en informatique, l'alphabet est constitué de deux symboles 0 et 1. Le français utilise quant à lui un peu plus de 26 symboles (car on peut ajouter aux 26 lettres de l'alphabet latin les caractères accentués et les signes de ponctuation). On appelle *source sans mémoire* une source pour laquelle la probabilité d'émission d'un symbole ne dépend pas des symboles précédemment émis.

Définition 7. *On appelle extension d'ordre k d'une source S la source S_k dont l'alphabet est obtenu par concaténation de k symboles consécutif de la source S.*

Évidemment, le débit de la source S_k est k fois moins élevé que celui de la source initiale S. De plus, si S est une source de symboles m-aire, l'alphabet de la source S_k sera de taille au plus m^k, cette valeur étant atteinte si S est sans mémoire.

[1] David Albert Huffman (9 août 1925 - 7 octobre 1999, Ohio),chercheur américain pionnier dans le domaine de l'informatique.

3.1.2 Entropie et mesure de la quantité d'information

Pour pouvoir compresser efficacement les symboles émis par une source, il est utile de savoir mesurer la quantité d'information apportée par les symboles qu'elle produit. Intuitivement, un symbole qui apparaît rarement, par exemple le "w" dans les textes en français, apporte plus d'information qu'un symbole très fréquent, par exemple le "e" toujours dans les textes en français[2].

Quantité d'information par symbole

Étant donné une source S émettant des symboles d'un alphabet A, on commence par calculer la probabilité d'apparition des symboles de A. Ce calcul peut-être fait *offline*, c'est-à-dire a posteriori, en calculant la fréquence d'apparition de chaque symbole si l'on dispose de l'ensemble du signal émis, ou bien *inline*, c'est-à-dire a priori, si l'on connaît certaines propriétés de la source, par exemple si l'on sait que la source émet des mots dans une certaine langue écrite.
Notons h la quantité d'information -que nous n'avons pas encore définie- apportée par un symbole $x \in A$. Faisons une liste des propriétés de h attendues.

1. Tout d'abord, on souhaite que la quantité d'information apportée par l'émission d'un symbole x augmente lorsque la probabilité d'émission de x, notée $p(x)$ dans la suite, diminue. Soit :

$$h(x) = f(\frac{1}{p(x)}),$$

   où f est un fonction croissante.

2. Si la probabilité d'émission d'un symbole est 1, alors celui-ci n'apporte aucune information. Soit :

$$f(1) = 0.$$

3. On souhaite enfin que la quantité d'information apportée par deux messages indépendants soit la somme des quantités d'information apportées par chacun des messages. Soit :

$$f\left(\frac{1}{p(x \text{ et } y)}\right) = f\left(\frac{1}{p(x).p(y)}\right) = f\left(\frac{1}{p(x)}\right) + f\left(\frac{1}{p(y)}\right).$$

Ces différentes propriétés impliquent que la fonction h doit être choisie de la forme :

$$h(x) = -\log_2(p(x)).$$

Le choix de la base 2 du logarithme sera expliqué plus loin.

Entropie d'une source

On souhaite maintenant définir la quantité moyenne d'information apportée par la source S. Cette quantité est appelée *entropie* et est naturellement définie par la formule :

$$H(S) := \sum_{x \in A} p(x)h(x) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

où l'on a noté p_i la probabilité d'émission du i-ème symbole de l'alphabet.

Maximisation de l'entropie

Notre but est maintenant de recoder les symboles de S de telle sorte que son entropie soit maximisée. Pour ce faire, on introduit le résultat préliminaire suivant.

[2] à l'exception de *La disparition* de G. Perec, un livre ne contenant pas -dans un but bien précis- la lettre "e".

Lemme 6. *(Inégalité de Gibbs) Soit n nombres p_i et q_i tels que :*

$$0 < p_i < 1, \qquad 0 < q_i < 1,$$

et

$$\sum_{i=1}^{n} p_i = 1, \qquad \sum_{i=1}^{n} q_i \leq 1.$$

Alors :

$$\sum_{i=1}^{n} p_i \log_2 \left(\frac{q_i}{p_i}\right) \leq 0,$$

avec égalité si et seulement si

$$\forall i,\ 0 \leq i \leq n, \qquad p_i = q_i.$$

Cette inégalité est également appelée *inégalité de Jensen*[3] dans d'autres contextes.
En appliquant ce lemme à l'entropie H, on déduit le résultat suivant :

$$H(S) \leq \log_2(m),$$

avec égalité dans le cas où tous les symboles de S sont équiprobables.

3.1.3 Propriétés d'un codage source

Comme précédemment annoncé, on va chercher à construire un nouvel alphabet de codage maximisant l'entropie, ou, du point de vue de la compression, minimisant la taille moyenne des nouveaux symboles obtenus :

$$\bar{\ell} = \sum_{i=1}^{m} n_i p_i,$$

où n_i est la taille du nouveau code du i-ème symbole de A.
On ajoute une autre contrainte sur le nouvel alphabet. On souhaite que celui-ci soit *auto-ponctué*[4], c'est-à-dire que les nouveaux symboles soient émis par simple concaténation. Ceci est formalisé par la définition suivante.

Définition 8. *On appelle code irréductible, ou code préfixe, ou code auto-ponctué, un ensemble de mots tel qu'aucun mot ne soit le début d'un autre.*

De la sorte, le récepteur d'un message sait toujours comment découper la suite de symboles qu'il reçoit, car il sait identifier les moments où l'on passe d'un symbole à un autre.
Tous les codes utilisés dans la vie courante ne sont pas irréductibles. Les langues écrites ne le sont pas en général, le code morse non plus par exemple.

Arbre $q-$aire

Les codes auto-ponctués sont de manière naturelle associés à la notion d'arbre q-aire, que l'on introduit maintenant.

Définition 9. *On appelle arbre q-aire un arbre dont chaque noeud est soit une feuille, soit possède q descendants.*

Par exemple un arbre binaire est obtenu en considérant l'arbre généalogique des ascendants d'une personne (et en se fixant une limite dans le temps). La notion d'arbre q-aire est le bon concept pour représenter les codes auto-ponctués. Le nombre q représente le nombre de caractères utilisés pour le nouveau codage des symboles de la source A et les mots sont donnés par l'ensemble des feuilles. Le découpage du message reçu se fait donc par parcours successifs de l'arbre, avec retour à la racine à chaque fois que l'on obtient une feuille. La figure 3.1 montre l'arbre de codage associé au code de 5 symboles présenté dans le tableau ci-dessous.
Les arbres q-aires vérifient en outre une certaine propriété, expliquée dans le lemme suivant.

[3] Johan Ludwig William Valdemar Jensen, surtout connu comme Johan Jensen, (8 mai 1859, Nakskov, Danemark - 5 mars 1925, Copenhague, Danemark) était un mathématicien et ingénieur danois. Il est surtout connu pour sa fameuse inégalité de Jensen. En 1915, il démontra également la formule de Jensen en analyse complexe.
[4] On parle aussi de code *préfixe*.

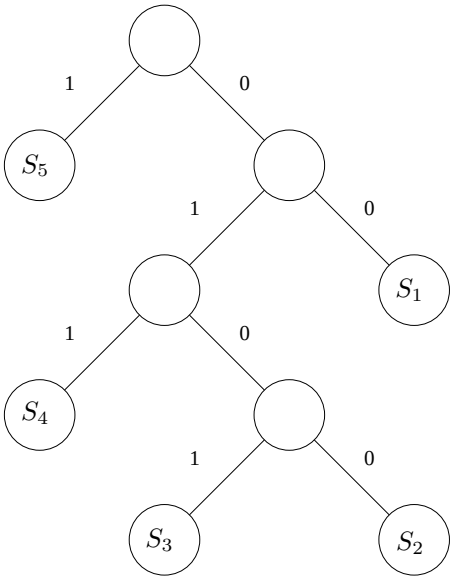

FIG. 3.1 – Arbre associé à un code auto-ponctué.

S_1	00
S_2	0100
S_3	0101
S_4	011
S_4	1

Lemme 7. *(Inégalité de Kraft[5] et réciproque) Les longueurs $(n_i)_{1\leq i\leq m}$ des m chemins allant vers les m feuilles d'un arbre q-aire vérifient :*

$$\sum_{i=1}^{m} q^{-n_i} \leq 1. \qquad (\textit{ inégalité de Kraft})$$

Réciproquement, si l'on se donne un entier q et m entiers $(n_i)_{1\leq i\leq m}$ vérifiant l'inégalité de Kraft, alors on peut construire un arbre q-aire ayant m feuilles situées à des profondeurs $(n_i)_{1\leq i\leq m}$.

Théorème fondamental du codage source

On dispose maintenant de tous les outils pour énoncer et démontrer le théorème fondamental du codage source, qui est également un théorème de Shannon.

Théorème 5. *Dans le codage d'une source S sans mémoire à l'aide d'un alphabet de taille q, la longueur moyenne $\bar{\ell}$ des mots du code utilisé pour coder un symbole de S vérifie :*

$$\frac{H(S)}{\log_2 q} \leq \bar{\ell},$$

et l'on peut approcher cette limite aussi près que l'on veut, quitte à coder les extensions de S au lieu de S elle-même.

Démonstration. On considère m symboles codés par des mots de longueurs $(n_i)_{1\leq i\leq m}$. Si ces mots sont obtenus comme les feuilles d'un arbre q-aire, alors leurs longueurs vérifient :

$$Q := \sum_{i=1}^{m} q^{-n_i} \leq 1.$$

[5]Cette inégalité fut publiée par Leon Kraft en 1949. Dans l'article correspondant Kraft ne considère que les codes auto-ponctués et attribue l'analyse qu'il présente pour obtenir son résultat à Raymond M. Redheffer. Cette inégalité est aussi parfois appelée "Théorème de Kraft-McMillan" après que Brockway McMillan l'ait découverte indépendamment en 1956. McMillan prouve le résultat pour une classe plus large de codes et attribue quant à lui la version correspondant aux codes auto-ponctués de Kraft à des observations de Joseph Leo Doob en 1955.

D'après l'inégalité de Gibbs, on a également :

$$\sum_{i=1}^{m} p_i \log_2 \left(\frac{q^{-n_i}}{p_i} \right) \leq 0,$$

où p_i est la probabilité d'émission du i-ème symbole de S et $q_i = \frac{q^{-n_i}}{Q}$, avec n_i la longueur du nouveau code du i-ème symbole. On en déduit :

$$-\sum_{i=1}^{m} p_i \log_2 p_i \leq \log_2 Q \times \sum_{i=1}^{m} p_i + \log_2 q \times \sum_{i=1}^{m} p_i n_i.$$

Or :
- $-\sum_{i=1}^{m} p_i \log_2 p_i = H(S)$,
- $\sum_{i=1}^{m} p_i = 1$,
- $\sum_{i=1}^{m} p_i n_i = \bar{\ell}$, la longueur moyenne d'un mot du nouveau code.

Finalement, on trouve :

$$H(S) \leq \bar{\ell} \log_2 q,$$

soit $H(S) \leq \bar{\ell}$ dans le cas d'un codage binaire. Nous avons donc obtenu la borne théorique de compression annoncée dans l'introduction de cette section.
Montrons maintenant qu'on peut s'approcher aussi près que l'on veut de cette borne.
On cherche à minimiser $\bar{\ell}$. Pour se faire, on peut essayer d'approcher le cas d'égalité dans les inégalités de Kraft et de Gibbs utilisées, c'est-à-dire à avoir :

$$Q = 1, \ \frac{q^{-n_i}}{Q} = p_i,$$

qui se traduit pour n_i par :

$$n_i = -\frac{\log_2 p_i}{\log_2 q}.$$

Mais cette dernière fraction n'est pas forcément égale à un entier, on choisit donc de définir n_i par :

$$-\frac{\log_2 p_i}{\log_2 q} \leq n_i \leq -\frac{\log_2 p_i}{\log_2 q} + 1.$$

En conséquence, on a :

$$\sum_{i=1}^{m} q^{-n_i} \leq \sum_{i=1}^{m} p_i = 1,$$

et l'inégalité de Kraft est vérifiée. Il existe donc un arbre q-aire correspondant aux longueurs n_i, dont on peut déduire les mots (de longueurs n_i) du nouveau codage. De plus on a :

$$-p_i \frac{\log_2 p_i}{\log_2 q} \leq n_i \leq -p_i \frac{\log_2 p_i}{\log_2 q} + p_i,$$

d'où l'on déduit par sommation :

$$\frac{H(S)}{\log_2 q} \leq \bar{\ell} \leq \frac{H(S)}{\log_2 q} + 1.$$

Pour approcher arbitrairement la limite théorique, une stratégie judicieuse est de considérer non plus la source S comme source initiale, mais son extension d'ordre k, S_k. En répétant le raisonnement précédent, on obtient :

$$\frac{H(S_k)}{\log_2 q} \leq \bar{\ell}_k \leq \frac{H(S_k)}{\log_2 q} + 1,$$

où $\bar{\ell}_k$ représente la longueur moyenne des mots du nouveau codage de S_k. Or :

$$H(S_k) = k.H(S),$$

et

$$\bar{\ell_k} = k\bar{\ell}.$$

On en déduit :

$$\frac{H(S)}{\log_2 q} \leq \bar{\ell} \leq \frac{H(S)}{\log_2 q} + \frac{1}{k}$$

□

Remarque 6. *Évidemment, approcher cette limite a un certain coût ! En effet, la technique de codage considérée implique à un moment ou à un autre la transmission d'un dictionnaire permettant le décodage pour retrouver les symboles émis initialement. Si l'on code les extensions de S au lieu de S elle-même, la taille du dictionnaire va augmenter et ce exponentiellement par rapport à l'extension considérée.*

3.1.4 Algorithme de Huffman

Il nous reste à donner une méthode permettant la mise en pratique du résultat théorème 5 (qui n'est pas constructif). C'est l'algorithme de Huffman qui fournit ce résultat à partir d'une source de m symboles, dont on connaît les probabilités $(p_i)_{1\leq i\leq m}$ d'émission. En pratique cet algorithme explique comment construire un *arbre de codage*.

Algorithme 1. *On effectue successivement les opérations suivantes :*

1. *Calcul ou évaluation de l'entropie de la source S et prescription d'un objectif de taille moyenne $\bar{\ell}$.*
2. *Classement des symboles d'une extension par probabilité d'apparition.*
3. *Regroupement dans la liste de q symboles de probabilités les plus faibles, et création d'un noeud. La probabilité de ce noeud est définie comme la somme des probabilités de ses q descendants.*
4. *Affectation des q symboles de l'alphabet de codage à chacune des branches du noeud obtenu à l'étape 3.*
5. *Itération des étapes 2 et 3, jusqu'à ce que la liste ne contienne qu'un élément.*
6. *Calcul des longueurs moyennes $\bar{\ell_k}$ et $\bar{\ell}$. Si l'objectif de longueur moyenne n'est pas atteint, on recommence à l'étape 2 avec une extension d'ordre plus élevé.*

Remarque 7. *Si m n'est pas multiple de $q-1$, on regroupe à la première itération de l'étape 3 non pas q symboles, mais $\tilde{q}$, où $\tilde{q}$ est choisi de telle sorte que : $m - \tilde{q} + 1$ soit un multiple de $q-1$.*

3.2 Codage canal et correction d'erreur

On considère maintenant uniquement des signaux binaires.
Une fois le signal – c'est-à-dire une suite de 0 et de 1 – compressé, il va maintenant s'agir de le transmettre. Le medium utilisé pour la transmission est appelé *canal*. Évidemment, les canaux peuvent-être de nature très variées. Il peut s'agir d'une onde électro-magnétique envoyée dans une fibre optique, dans l'atmosphère ou dans dans l'espace dans le cas de la communication par satellite, d'impulsions électriques envoyées dans un réseau, ou encore d'une clef USB, d'un disque dur ou d'un disque optique codé sous le format `.wav` pour les CD ou sous des formats plus évolués pour les DVD.
Chacun de ces canaux rend intrinsèquement possible l'introduction d'erreurs. Il y a concrètement toujours un risque que le médium transforme un 0 en 1 et réciproquement. Cela peut ne pas avoir d'effet grave si le code supporte une image ou un son, mais peut être beaucoup plus lourd de conséquence s'il s'agît du texte d'un programme exécutable par exemple. Dans ce cas, la transmission échouera si le moindre symbole du programme est modifié.
Il apparaît donc nécessaire de mettre en place une stratégie permettant au moins de détecter, lors de la réception, les erreurs de transmission, voir de les corriger. De nombreuses méthodes atteignant cet objectif ont été conçues depuis une quarantaine d'années. Elles reposent toutes sur l'ajout de *bits de correction*, ou *bits de contrôle*, c'est-à-dire qu'elles ont toujours pour effet négatif de grossir la taille du signal à envoyer. Un compromis doit donc être trouvé entre qualité de correction et alourdissement du signal.
Dans cette section, on présente des outils permettant de quantifier le facteurs intervenant dans ce compromis

et une stratégie générale de codage canal ainsi qu'une de ses réalisations concrètes.
Dans la suite on désignera indifféremment de *code correcteur*, *codage canal* ou encore *code* la stratégie de correction considérée.

3.2.1 Une approche naïve

Une idée simple que l'on peut avoir est de répéter un certain nombre de fois chaque symbole à transmettre. Si on choisit par exemple de doubler le symbole, c'est-à-dire d'appliquer la fonction :

$$0 \to 00$$

$$1 \to 11,$$

on pourra facilement détecter une erreur (si les symboles reçus ne coïncident pas deux à deux). Par contre, on ne pourra corriger l'erreur correctement qu'une fois sur deux en moyenne - en choisissant arbitrairement de remplacer la séquence erronée détectée par un 0, par exemple.
Dans cette démarche, on a ajouté un bit de correction par bit transmis. Si on choisit maintenant d'en ajouter deux, en adoptant une stratégie de triplement, on constate que la correction sera plus efficace : on remplacera une séquence erronée, par exemple '001', par le signe apparaissant majoritairement[6] dans le triplet, 0 dans notre exemple. Cette méthode permet de détecter deux erreurs par bloc de trois bits et d'en corriger une : en effet, si deux erreurs affectent, lors de la transmission la séquence de trois bits, alors la stratégie choisie donnera lieu, lors du décodage, à une erreur. Pour aller plus loin, on peut regarder l'exemple d'une canal introduisant 5% d'erreurs, avec lequel on applique la stratégie du triplement. Puisqu'une erreur est corrigée, la probabilité qu'une séquence de trois bits soit transmise correctement ou avec une seule erreur est de :

$$p = 0,95^3 + 3 \times 0,95^2 \times 0,005 = 0,99275.$$

Ainsi le taux de succès est passé de 95% à un taux supérieur 99%. Évidemment, ceci a un coût ! Il a en effet fallu tripler la taille du signal à transmettre.
On va voir dans la suite comment optimiser l'usage des bits de correction.

3.2.2 Codes linéaires par blocs

Découpage en bloc et structure algébrique des blocs

Un préalable souvent nécessaire à la correction d'erreur est le découpage du signal à transmettre en blocs de taille fixe, disons k. L'ensemble sur lequel on va travailler est donc B^k où $B = \{0, 1\}$. En munissant celui-ci de l'addition composante par composante dans le groupe $(\mathbb{Z}/2\mathbb{Z}, +)$, ainsi que du corps des scalaires (fini) $(\mathbb{Z}/2\mathbb{Z}, +, \times)$, on voit que l'on peut doter B^k d'une structure d'espace vectoriel (fini).

Principe du codage linéaire et notations

Dans le cadre que l'on vient d'introduire, l'ajout de r bits de correction peut être vu comme l'application d'une certaine fonction injective :

$$g : B^k \to B^n,$$

où $n = k + r$. La stratégie de correction est dite *linéaire*, lorsque g est une application linéaire (entre les espaces B^k et B^n). Si on note m un mot de taille k et m' son image par le codage, on a donc :

$$m' = Gm,$$

où G est la matrice de g de dimensions (m, k) et à coefficients dans $\mathbb{Z}/2\mathbb{Z}$.
Puisque $n > k$, l'image de g est un sous-espace vectoriel de B^n. On le note dans la suite $C(k, n)$. En tant que sous-espace de dimension k, il est caractérisé par $n - k = r$ équations linéaires traduisant les dépendances entre les bits des blocs codés. Ces relations peuvent s'écrire sous la forme matricielle :

$$m' \in C(k, n) \Leftrightarrow Hm' = 0,$$

où H est une matrice de dimensions (r, n).

[6] On parle parfois de *stratégie du vote majoritaire*.

Notion de syndrome

Comme stipulé précédemment, la transmission à travers le canal peut donner lieu à une erreur affectant certains bits du mot. Dans le formalisme précédent, le mot m^* recueilli en sortie de canal sera donc de la forme :

$$m^* = m' + e,$$

où e est un vecteur de B^n comportant des 1 sur les composantes correspondant à celles affectées et des 0 ailleurs. On a alors la relation :

$$Hm^* = He,$$

puisque $m \in KerH$. Le vecteur He est appelé *syndrome*. Si une erreur s'est produite sur la composante i de m', alors le syndrome sera égal à la i-ème colonne de H. On appelle généralement *matrice de contrôle* la matrice H.

3.2.3 Détection et correction d'erreur

On continue notre formalisation, en s'intéressant maintenant plus précisément à l'ensemble des mots du code, c'est-à-dire $C(k,n)$.

Distance de Hamming

Dans la formalisation que nous mettons petit à petit en place, il est utile de comparer les mots 2 à 2 puisque cela permet de décider si un mot reçu appartient ou non à $C(k,n)$. La notion suivante permet une comparaison efficace.

Définition 10. *(distance de Hamming[7]) Soit m_1 et m_2 de B^n. On défini la distance d entre m_1 et m_2 comme étant le nombre de bits différents entre les deux vecteurs.*

On appelle cette distance *distance de Hamming*. On a par exemple :

$$d(111, 101) = 1.$$

Une propriété intéressante de cette distance est qu'elle vérifie :

$$d(m_1, m_2) = d(m_1 + m_2, 0_{B^n}), \tag{3.1}$$

où :

$$0_{B^n} = {}^t(\underbrace{0, ..., 0}_{n\ fois}).$$

Autrement dit, pour connaître la distance entre deux mots, il suffit de les additionner et de compter le nombre de composantes égales à 1 dans le résultat.
Derrière cette notion de distance, se cache une stratégie de correction : étant donné que les mots du code ne forment qu'une sous-partie de B^n, lorsqu'on reçoit en sortie de canal un mot ne figurant pas dans $C(k,n)$, on le remplace par le mot du code le plus proche.

Décodage par maximum de vraisemblance

On introduit encore quelques notions permettant cette fois-ci de décrire $C(k,n)$ et de faire le lien avec la correction d'erreur.

Définition 11. *On appelle distance d'un code (de correction linéaire par bloc) la plus petite distance entre deux mots de $C(k,n)$.*

Le lemme suivant donne une méthode simple pour calculer la distance d'un code.

[7] Richard Wesley Hamming (11 février 1915, Chicago - 7 janvier 1998, Monterey, Californie) est un mathématicien américain surtout connu pour son algorithme de correction d'erreur, le Code de Hamming. Il travailla avec Claude Shannon entre 1946 et 1976 aux laboratoires Bell.

Lemme 8. *La distance d'un code est égale au nombre de 1 contenus dans le mot non nul du code qui en contient le moins.*

Démonstration. Ce lemme est une conséquence directe de la formule (3.1). En effet, puisque $C(k,n)$ est un espace vectoriel, on a :

$$C(k,n) - \{0_{B^n}\} = \{m_1 + m_2 / m_1 \neq m_2,\ m_1 \in C(k,n),\ m_2 \in C(k,n)\}\,.$$

□

Si la distance d'un code est 1, un mot avec un bit erroné peut également être un mot du code. On ne pourra donc pas à coup sûr détecter 1 erreur. Si la distance est 2, on détectera à coup sûr une erreur, mais on ne pourra pas la corriger dans tous les cas. En effet, il se peut que le mot entaché d'une erreur soit équidistant de deux mots distincts du code, ce qui fait qu'on ne pourra choisir qu'arbitrairement et sans garantie le mot du code qui lui correspondait avant l'erreur. Enfin, si la distance est 3, alors, par un raisonnement analogue, on pourra détecter 2 erreurs et corriger correctement 1 erreur.
Tout ceci se généralise dans le lemme suivant.

Lemme 9. *Pour pouvoir détecter t erreurs, la distance d'un code doit au moins être égale à $t+1$. Pour pouvoir corriger t erreurs, la distance d'un code doit au moins être égale à $2t+1$.*

Les figures 3.2 et 3.3 illustrent ce lemme.

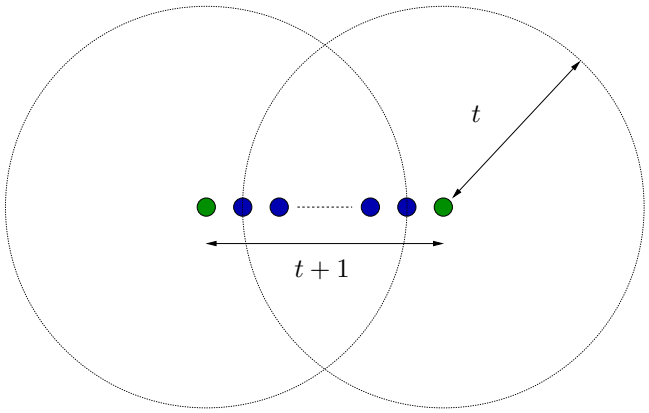

FIG. 3.2 – Détection d'erreur : les boules de rayon t ne contiennent qu'un mot, par contre les mots qu'elles contiennent appartiennent à plusieurs boules.

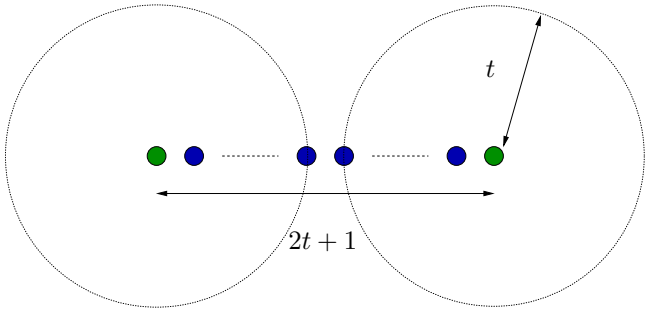

FIG. 3.3 – Correction d'erreur : les boules de rayon t ne contiennent qu'un mot et les mots qu'elles contiennent n'appartiennent qu'à une boule.

3.2.4 Codes de Hamming

Pour achever la description de la stratégie de correction, il faut spécifier les matrice G et H. C'est l'objet de cette section.

Choix des paramètres

On se place dans le cadre simple où au plus une erreur peut affecter les mots (de longueur n) du code. Il y a donc $n+1$ scenari possibles : soit il n'y a pas eu d'erreur pendant la transmission, soit une erreur s'est produite sur l'un des n bits. Or, d'après les dimensions des espaces considérés, 2^r syndromes possibles. Si l'on veut pouvoir faire correspondre de manière sûre, c'est-à-dire par le biais d'une injection, les vecteurs syndromes observés et le scénario dont il est la conséquence, il faut nécessairement :

$$2^r \geq n+1.$$

Puisque l'on souhaite optimiser la taille des mots du code, on va chercher à choisir le n le plus petit possible. Le meilleur résultat est donc obtenu lorsque :

$$2^r = n+1. \tag{3.2}$$

D'autre part, on a :

$$n = k+r. \tag{3.3}$$

Enfin $k \geq 1$, car on travaille sur des blocs de longueur non nulle ! En cherchant quels sont les k pour lesquels (3.2) et (3.3) ont des solutions (k,n) entières, on trouve par exemple :

$$k=1,\ n=3,\ r=2,$$

et ensuite :

$$k=4,\ n=7,\ r=3.$$

Le premier exemple correspond à la stratégie de triplement, décrite à la section 3.2.1. Le second correspond à un code largement utilisé, souvent appelé $C(4,7)$, ce qui correspond aux notations utilisées dans cette section.

Optimisation du syndrome

Dans cette dernière section, on explicite le code $C(4,7)$. Pour être facile à utiliser, il est utile de faire en sorte que le syndrome représente l'équivalent binaire de l'emplacement de l'erreur. Dans ce cas, la matrice H est donnée par :

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Par exemple, si une erreur se produit sur le troisième bit de m', alors :

$$e = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

et donc :

$$He = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix},$$

ce qui correspond bien à 3 en écriture binaire.
Déterminons maintenant le noyau de H. En notant $(c_i)_{1\leq i\leq 7}$ les 7 composantes d'un mot m', on a :

$$m' \in Ker\ H \Rightarrow \left\{ \begin{array}{rcl} c_4 + c_5 + c_6 + c_7 & = & 0 \\ c_2 + c_3 + c_6 + c_7 & = & 0 \\ c_1 + c_3 + c_5 + c_7 & = & 0 \end{array} \right. ,$$

que l'on peut par exemple résoudre en :

$$\begin{array}{rcl} c_1 & = & c_3 + c_5 + c_7 \\ c_2 & = & c_3 + c_6 + c_7 \\ c_4 & = & c_5 + c_6 + c_7 \end{array} .$$

Dans ce cas, les 4 bits à coder sont c_3, c_5, c_6, c_7 et les bits de contrôle sont c_1, c_2, c_4.
On peut montrer que pour un canal ayant une probabilité d'erreur de 5%, la probabilité de transmettre correctement 4 bits est $.95^4 \approx 81\%$. L'utilisation de $C(4,7)$ permet de faire passer cette valeur à $(1-p)^7+7p(1-p)^6 \approx 95\%$. Ce résultat, comparable à la stratégie par triplement, est en fait bien meilleur, car la taille des mots du code n'a même pas doublé.

Remarque 8. *Notons enfin qu'on peut rendre la probabilité d'erreur aussi faible que l'on veut en appliquant un code correcteur plusieurs fois de suite.*

3.3 Notes bibliographiques

Ce chapitre s'inspire de notes de cours gracieusement fournies par Yvan Pigeonnat. Pour plus d'informations sur le codage source, on pourra consulter les références [2], [4]. Pour plus d'informations sur les codes correcteurs (de type Hamming), on pourra également consulter [4], mais bien d'autres documents relatifs à la correction d'erreur se trouvent facilement sur Internet.

Chapitre 4

Transformation de Fourier discrète

Ce chapitre est un préliminaire aux chapitres qui suivent. Avant d'aborder les filtres, qui permettent de travailler et de modifier la transformée de Fourier, en d'autres termes le *spectre* d'un signal, il est nécessaire d'indiquer comment calculer effectivement ce spectre. Puisque les signaux auxquels on s'intéresse sont discrets (car numériques), on se concentre ici sur la transformée de Fourier discrète. Son calcul, si il est fait naïvement, peut être très coûteux. Il peut cependant être accéléré considérablement en utilisant une méthode due à Tuckey[1] et Cooley[2], le fameux algorithme *FFT* , dont le nom est l'acronyme correspondant à *Fast Fourier Transform*.

4.1 La TFD

On commence par rappeler le cadre discret où l'on se place.

4.1.1 Cadre et problèmes

On considère un signal échantillonné, fini :

$$s = (s_0, s_1, ..., s_{N-2}, s_{N-1}),$$

obtenu à partir d'un signal continu[3], également noté s, par une formule du type :

$$s_n = s(n.\Delta t),$$

où Δt est le pas de l'échantillonnage, relié à la fréquence d'échantillonnage $2B$ du chapitre 1 par l'équation :

$$\Delta t = \frac{1}{2B}.$$

On ne tient pas compte ici du fait que le signal peut avoir été quantifié, c'est-à-dire que les s_n sont considérés a priori comme réels.

Définition 12. *On appelle Transformée de Fourier Discrète, en abrégé TFD, la suite finie*

$$\widehat{s} = (\widehat{s}_0, \widehat{s}_1, ..., \widehat{s}_{N-2}, \widehat{s}_{N-1})$$

définie par la formule :

$$\widehat{s}_k := \sum_{n=0}^{N-1} s_n \mathrm{e}^{-2i\pi k \frac{n}{N}}.$$

[1] James Cooley (1926- .) est un mathématicien americain.

[2] John Wilder Tukey (16 juin 1915, New Bedford, Massachusetts - 26 juillet 2000, New Brunswick, New Jersey.) était un statisticien américain.

[3] Autrement dit analogique.

Comme toute Transformée de Fourier, on voit que la TFD est une application linéaire. Sa matrice est une *matrice pleine*, c'est-à-dire ne comportant pas de 0, de type matrice de Vandermonde.
On voit de plus que le pas de temps utilisé n'apparaît pas dans la formule. On peut en quelque sorte considérer que celui-ci est égal à 1. Ceci est en fait naturel : un signal discret est une suite de nombres et seules des informations supplémentaires sur ce qu'elle représente permettent de connaître la fréquence qui a été utilisée pour obtenir l'échantillon.
On va s'intéresser à deux questions dans la suite :

1. Quel est le coût de calcul de la TFD ? et comment l'améliorer ?
2. Quelle est la qualité de l'approximation de la transformée de Fourier du signal analogique initial ?

4.1.2 Propriétés de la TFD et signaux périodiques discrets

Pour simplifier l'exposé, on introduit de nouvelles notations.
Soit une suite finie $a = (a_0, a_1, ..., a_{N-2}, a_{N-1})$, et $\omega_N = \mathrm{e}^{-i\frac{2\pi}{N}}$. Alors la suite $A = (A_0, A_1, ..., A_{N-2}, A_{N-1})$, définie par

$$A_m = \sum_{n=0}^{N-1} a_n \omega_N^{nm} \tag{4.1}$$

est la TFD de a.

Théorème 6. *(Formule d'inversion de la TFD) En gardant les notations précédentes, on a :*

$$a_n = \frac{1}{N} \sum_{m=0}^{N-1} A_m \omega_N^{-nm}.$$

Évidemment, on retrouve dans la TFD et sa formule d'inversion les analogies habituelles entre les transformées de Fourier et leurs inverses. Dans le cas présent, on retrouve simplement que l'inverse d'une matrice de Vandermonde est une matrice de Vandermonde[4].
Cette formule reste valable si on considère les extensions périodiques de a et A, définies par les formules :

$$a_{n+kN} = a_n, \quad A_{m+kN} = A_m.$$

C'est maintenant comme cela que l'on envisagera les choses. Tous les signaux considérés seront vus comme des signaux discrets périodiques. Par conséquent, l'ordre de sommation peut-être choisi arbitrairement. En supposant par exemple que $N = 2M + 1$, on obtient :

$$A_m = \sum_{n=0}^{N-1} a_n \omega_N^{nm}, \quad a_n = \frac{1}{2M+1} \sum_{m=-M}^{M} A_m \omega_N^{-nm}.$$

Pour simplifier encore, on notera dans la suite ces relations en abrégé par :

$$(a_n) \underset{N}{\longleftrightarrow} (A_m).$$

Théorème 7. *Si* $(a_n) \underset{N}{\longleftrightarrow} (A_m)$ *et* $(b_n) \underset{N}{\longleftrightarrow} (B_m)$, *alors :*

$$\Big(\sum_{k=0}^{N-1} a_k b_{n-k}\Big) \underset{N}{\longleftrightarrow} (A_m B_m).$$

Ce théorème est une variante de ceux que nous avions déjà vus au chapitre 1 sur le lien entre séries de Fourier et convolution.

[4] proportionnelle à une matrice de Vandermonde, pour être plus précis.

4.2 L'algorithme FFT

On commence par constater qu'une approche directe n'est pas vraiment efficace.

4.2.1 Implémentation naïve

En appliquant la formule (4.1), on est conduit à calculer la somme suivante :

$$A_m = a_0 + a_1\omega_N^m + a_2\omega_N^{2m} + ... + a_{N-1}\omega_N^{m(N-1)}.$$

Si on ne retient que les multiplications et en considérant que la suite $(\omega_N^{km})_{k=0,...,N-1}$ a été pré-calculée[5], on est conduit à effectuer $N-1$ opérations. Le calcul complet de la suite A nécessite donc $(N-1)^2$ opérations, ce qui n'est pas satisfaisant du tout[6].

4.2.2 L'algorithme de Tuckey et Cooley

Publié en 1965 dans un article devenu depuis célèbre, Tuckey et Cooley ont donné une méthode permettant de réduire considérablement le temps de calcul de la suite A précédente.

Un exemple

On va montrer comment réduire à $N\log_2 N$ le nombre de multiplications. On se place dans le cas où la taille des suites considérées est paire et on considère

$$(a_n) \underset{2N}{\longleftrightarrow} (A_m).$$

On introduit alors deux suites $(b_n)_{n=0...N-1}$ et $(c_n)_{n=0...N-1}$ définies par $b_n = a_{2n}$ et $c_n = a_{2n+1}$ et on note :

$$b_n \underset{N}{\longleftrightarrow} B_m, \quad c_n \underset{N}{\longleftrightarrow} C_m.$$

On remarque alors que :

1. $A_m = B_m + \omega_{2N}^m C_m$, pour $m = 0, ..., 2N-1$,
2. $B_{m+N} = B_m$, $C_{m+N} = C_m$ et $\omega_{2N}^{m+N} = -\omega_{2N}^m$.

On en déduit :

$$A_m = B_m + \omega_{2N}^m C_m \qquad m = 0, ..., N-1 \tag{4.2}$$
$$A_{m+N} = B_m - \omega_{2N}^m C_m \qquad m = N, ..., 2N-1. \tag{4.3}$$

On a vu que le calcul de B et C nécessitait $(2N-1)^2$ opérations. Le calcul de A peut alors être fait à partir de B et C par $N-1$ opérations supplémentaires. De plus, on voit qu'ayant calculé tous les termes (4.2), on dispose sans calcul supplémentaire de tous les termes (4.3).
Le coût total est donc :

$$2(N-1)^2 + N - 1 = (N-1)(2N-3),$$

au lieu de $(2N-1)^2$, ce qui fait qu'on a divisé le temps de calcul par, grosso-modo, 2.

[5] c'est-à-dire calculée une fois pour toute, en supposant que l'entier N est fixé.

[6] L'objectif de tout créateur d'algorithme est en effet souvent d'obtenir des coût de calcul de l'ordre de N, ou, à défaut, de l'ordre de $N\log(N)$. C'est par exemple ce qui a motivé l'introduction du fameux algorithme de tri rapide *"Quicksort"*.

Cas général

Mais on peut aller encore plus loin ! Il suffit pour cela d'appliquer récursivement le raisonnement précédent. Supposons pour ce faire que $N = 2^s$ et notons $F(N)$ le nombre d'opérations nécessaire au calcul de la TFD d'une suite de taille N, c'est-à-dire d'un signal N-périodique.

Théorème 8. *Il existe un algorithme de calcul de TFD vérifiant :*

$$F(N) \leq \frac{1}{2} N \log_2 N.$$

Démonstration. En appliquant la méthode présentée dans l'exemple, on obtient :

$$F(2N) = 2F(N) + N - 1 \leq 2F(N) + N.$$

Supposons que $F(N) \leq \frac{1}{2} N \log_2 N$, alors

$$2F(N) + N \leq N(\log_2 N + 1) = \frac{1}{2} 2N \log_2(2N),$$

ce qui achève la démonstration par récurrence. □

Pour illustrer ce résultat, considérons une suite de $N = 1024$ termes. Par la méthode naïve expliquée en introduction de cette section, le calcul nécessite $(N-1)^2 = 1\ 046\ 529$ opérations. Par l'algorithme FFT de Tuckey et Cooley, il nécessite moins de $\frac{1}{2} N \log_2 N = 5120$ opérations. Le rapport du temps de calcul entre les deux méthodes est donc d'à peu près 200. Autrement dit, ce qui nécessitait presque 7 mois ne requiert maintenant qu'une journée.
Si la taille de l'échantillon n'est pas une puissance de 2, ce qui n'arrive pas en traitement numérique du signal, on complète généralement l'échantillon par des zéros pour pouvoir appliquer l'algorithme.

4.3 Qualité de l'approximation

On se donne maintenant un pas de temps Δt. Commençons par rappeler la formule de Poisson, version forte :

$$\forall \nu \in \mathbb{R}, \quad \sum_{n \in \mathbb{Z}} s(n\Delta t) \mathrm{e}^{-2i\pi \Delta t \nu} = \frac{1}{\Delta t} \sum_{n \in \mathbb{Z}} \widehat{s}(\nu + \frac{n}{\Delta t}).$$

On va voir une deuxième application de cette formule.

Théorème 9. *Soit $M \in \mathbb{N}$ et s un signal de support inclus dans $[-M\Delta t, M\Delta t]$. On a :*

$$\sum_{n=_M}^{M} s(n\Delta t) \mathrm{e}^{-2i\pi \frac{kn}{2M+1}} = \frac{1}{\Delta t} \sum_{n \in \mathbb{Z}} \widehat{s}(\frac{n}{\Delta t} + \frac{k}{(2M+1)\Delta t}).$$

On voudrait que le terme de gauche représente $\widehat{s}(\frac{k}{(2M+1)\Delta t})$, l'erreur est donc :

$$\sum_{n \in \mathbb{Z}^*} \widehat{s}(\frac{n}{\Delta t} + \frac{k}{(2M+1)\Delta t}).$$

Une fois encore cette erreur est irréductible, comme c'était le cas pour la série infinie obtenue dans la formule de reconstruction (1.6). On peut cependant la contrôler[7], en jouant sur le paramètre Δt pour que $\widehat{s}$ soit négligeable en dehors de $[-\frac{1}{2\Delta t}, \frac{1}{2\Delta t}]$.

[7] de la même manière que l'on pouvait sur échantillonner le signal pour mieux le reconstruire

4.4 Une application au produit de polynômes

Cette dernière section est plus anecdotique, car elle n'entre pas directement dans le cadre du traitement numérique du signal. Elle permet cependant de voir que l'algorithme FFT a une portée plus large que le simple calcul de transformée de Fourier discrète.

4.4.1 Notations

Supposons que l'on souhaite calculer le produit de deux polynômes, de coefficients $a_0, ..., a_{N-1}$ et $b_0, ..., b_{N-1}$. Notons $c_0, ..., c_{2N-2}$ les coefficients du polynôme produit.

4.4.2 Représentation des polynômes

La représentation des polynômes par leur coefficients a beau être la plus répandue, elle n'est pas universelle. Avant de voir quelles en sont les alternatives, rappelons quelques-unes de ses propriétés.

Représentation par coefficient

Un des intérêts de la représentation par coefficients est que le coût du calcul de la valeur du polynôme en un point est de l'ordre de N, en utilisant l'algorithme de Hörner. Par contre, cette représentation n'est pas du tout pratique pour effectuer le calcul du produit de deux polynômes. Il faut en effet calculer $N-1$ produits de convolution, ce qui revient à un coût de calcul de l'ordre de $(N-1)^2$.

Représentation par point-valeur

On introduit maintenant une autre représentation des polynômes, moins utilisée que la précédente mais qui possède des avantages significatifs pour la multiplication.
On peut également représenter un polynôme de degré $N-1$ par la donnée de N couples (x_n, y_n) avec

$$n \neq n' \Rightarrow x_n \neq x_{n'}, \tag{4.4}$$

et où y_n représente la valeur du polynôme au point y_n. Cette représentation des polynômes, appelée *représentation par point-valeur*, est bien injective de l'ensemble des polynômes de degrés $N-1$ dans $\left(\mathbb{R}^2\right)^{N-1}$, sous réserve que le choix de la suite $(x_n)_{n=0,...,N-1}$ vérifie (4.4).
On voit tout de suite que multiplier des polynômes représentés en point-valeur consiste simplement en N multiplications, ce qui représente un gain significatif par rapport à la représentation par coefficients. Par contre l'évaluation en un point (différent des x_n, sinon, c'est immédiat) est moins performante. On peut par exemple la faire en appliquant une formule d'interpolation de Lagrange, mais celle-ci nécessite un calcul de l'ordre de N^2.

4.4.3 Utilisation de la FFT

Expliquons maintenant comment réaliser avec un coût de calcul de l'ordre de $N \log_2 N$ le produit de deux polynômes en représentation par coefficients.
L'idée consiste simplement à passer les deux polynômes à multiplier en représentation point-valeur, en choisissant pour suite $(x_n)_{n=0,...,N-1}$ les racines $2N$-ième de l'unité. On voit immédiatement que le calcul de N point-valeur revient dans ce cas au calcul des TFD des suites $a_0, ..., a_{N-1}$ et $b_0, ..., b_{N-1}$. En utilisant l'algorithme FFT, ceci peut être effectué avec un coût de calcul de l'ordre de $N \log_2 N$.
Étant maintenant en représentation point-valeur, le polynôme produit dans cette représentation est calculé en $2N$ opérations. Il reste finalement à revenir en représentation par coefficients, ce qui se fait une nouvelle fois en appliquant l'algorithme FFT, avec un coût de l'ordre de $N \log_2 N$.

4.5 Notes bibliographiques

Ce chapitre s'inspire de la présentation de la FFT faite dans la référence [1], qui ne contient pas l'application aux polynômes. On pourra également consulter les chapitres 7 et 8 de la référence [3] pour une autre présentation de l'algorithme, moins mathématique. Ici encore, de nombreux textes et documents concernant la FFT sont disponibles sur Internet.

Chapitre 5

Filtres numériques

On aborde dans ce chapitre le dernier grand volet de ce cours, le filtrage des signaux numériques. Une fois le signal échantillonné et quantifié, il est utile pour de nombreuses applications d'en définir et calculer le contenu fréquentiel, c'est-à-dire son spectre. L'algorithme de la FFT nous permet de réaliser efficacement cet objectif. L'étape suivante, à laquelle on s'attaque maintenant, est d'agir sur ce spectre de manière à en atténuer, amplifier, sélectionner ou occulter certaines fréquences, ou certaines plages de fréquences. Cet démarche s'appelle le *filtrage* et peut être réalisée, pour ce qui concerne les signaux discrets, par des *filtres numériques*[1]. Dans ce chapitre, on donne les bases permettant de définir un filtre et ses propriétés.

5.1 Filtres linéaires

Un filtre numérique peut-être vu de plusieurs manières, selon le domaine dans lequel on travaille. Certains le définiront par un circuit électronique, d'autres par un assemblage de portes logiques ... Notre penchant pour les mathématiques nous conduit plutôt à les assimiler à des algorithmes de calculs. Mais on parle en tout cas de la même chose.

5.1.1 Définitions

On donne donc une définition mathématique d'un filtre numérique.

Définition 13. *Un filtre numérique est une application de* $\ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$.

Dans la pratique, les signaux rencontrés sont finis, si bien que le fait de travailler dans $\ell^2(\mathbb{Z})$ plutôt que dans un autre $\ell^p(\mathbb{Z})$ n'a pas beaucoup d'importance. Le fait de choisir cet espace plutôt qu'un autre vient du fait que la norme 2 d'un signal (discret ou pas) est souvent égale à son énergie et que d'un point de vue physique, il est raisonnable de ne considérer que des signaux d'énergie finie.
Un filtre numérique est donc un système qui produit un signal discret[2], que l'on notera dans la suite $y = (y_n)_{n\in\mathbb{Z}}$ à partir d'un signal reçu en entrée $x = (x_n)_{n\in\mathbb{Z}}$. On appelle parfois *excitation* ou *entrée* du filtre la suite x et *réponse* ou *sortie* du filtre la suite y.

Remarque 9. *(IMPORTANTE) Dans toute la suite, on ne considère – sauf mention contraire – que des filtres qui élaborent la réponse en un temps fixé n en fonction d'un nombre fini de termes de x ou de y.*

Dans la réalité, les dispositifs dont on dispose ne permettent de toute façon que de travailler dans le cadre fixé par cette remarque.
Donnons un exemple de filtre. La formule :

$$y_n = x_{n-1} \tag{5.1}$$

[1] Il existe aussi des filtres analogiques, qui agissent sur des signaux continus en temps. On en trouve dans la nature, par exemple l'oreille humaine, ou plus généralement tout système physique répondant à une excitation. Du point de vue de l'activité humaine, l'essor massif des technologies numériques les place plutôt sur la liste des espèces en voie de disparition.
[2] une suite donc.

définit un filtre qui effectue un décalage unitaire. On fera souvent appel dans la suite à ce filtre élémentaire, si bien qu'on fixe d'ores-et-déjà sa notation.

Définition 14. *On appelle décalage unitaire et on note τ le filtre défini par la relation* (5.1).

Filtres récursifs

Mais la suite y peut être définie de manière plus compliquée, par exemple de manière récurrente, comme c'est le cas pour :

$$y_n = \frac{1}{4}y_{n-1} + \frac{1}{2}x_n + \frac{1}{2}x_{n-1}. \tag{5.2}$$

On parle alors de *filtre récursif*. Attention, un même filtre peut avoir une définition récursive et une définition non récursive. Par exemple :

$$y_n = \frac{1}{2}y_{n-1} + x_n$$

produit le même filtre que :

$$y_n = \sum_{k\in\mathbb{N}} (\frac{1}{2})^k x_{n-k}.$$

Par contre, cette dernière formule n'entre pas dans le cadre des filtres non-récursifs que l'on considère dans la suite de ce cours, puisqu'on s'est placé dans le champs de la remarque 9.

Représentation en schéma-bloc

Il est très pratique de représenter les filtres sous forme de schéma-bloc. C'est à la fois très intuitif, facile à former à partir de l'équation de définition du filtre et proche de ce qui serait un circuit électronique réalisant effectivement le filtre. Les figures 5.1 et 5.2 sont des représentations sous forme de schéma-bloc du filtre décrit par l'équation (5.2). La notation z^{-1} sera définie dans la suite.

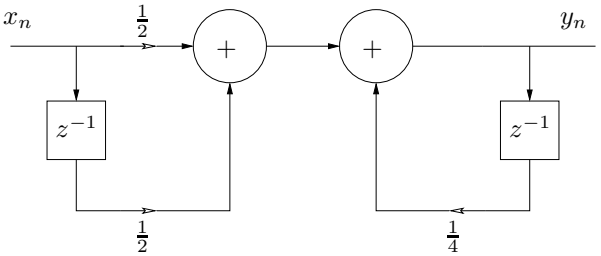

FIG. 5.1 – Une représentation du filtre défini par (5.2).

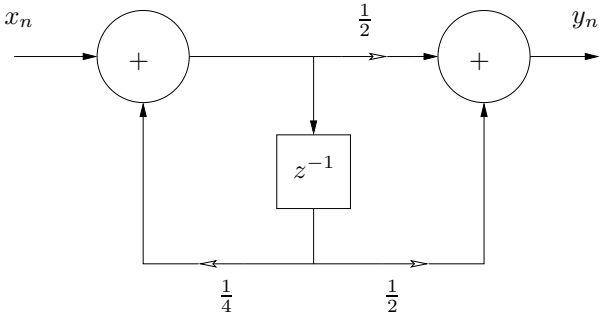

FIG. 5.2 – Une autre représentation du même filtre.

Réponse impulsionelle

Dans la suite, on considérera souvent l'image par les filtres de l'entrée *impulsion*, c'est-à-dire le signal spécifique :

$$\delta^0_n = \begin{cases} 1 & \text{si} \quad n = 0, \\ 0 & \text{sinon.} \end{cases}$$

Définition 15. *(Réponse impulsionelle) On appelle réponse impulsionnelle d'un filtre numérique l'image par ce filtre du signal d'entrée $\delta^0 = (\delta^0_n)_{n\in\mathbb{Z}}$.*

Dans la suite, la réponse impulsionelle sera notée $h = (h_n)_{n\in\mathbb{Z}}$.

5.1.2 Propriétés des filtres

On passe maintenant en revue les premières propriétés des filtres numériques. Pour simplifier les définitions, on note $\mathcal{F}$ le filtre, considéré comme une fonction.

Définition 16. *(Invariance temporelle) Un filtre est dit invariant en temps, si l'image par $\mathcal{F}$ de la suite décalée $(x_{n-n_0})_{n\in\mathbb{Z}}$, où n_0 est un entier relatif fixé, est la suite décalée de $y=(y_n)_{n\in\mathbb{Z}}$, c'est-à-dire $(y_{n-n_0})_{n\in\mathbb{Z}}$.*

De manière équivalente, un filtre est invariant en temps si sa fonction $\mathcal{F}$ commute avec la fonction décalage τ. Par exemple la relation :

$$y_n = nx_n,$$

ne définit pas un filtre invariant en temps. Tous les filtres que nous considérerons à partir de maintenant seront invariants en temps.

Définition 17. *(Linéarité) Un filtre est dit linéaire si la relation de dépendance de y à x est linéaire.*

Autrement dit, un filtre est dit linéaire si la fonction $\mathcal{F}$ est une application linéaire.
Un premier théorème permet de caractériser simplement les filtres linéaires invariants en temps.

Théorème 10. *Un filtre linéaire invariant en temps est entièrement déterminé par la donnée de sa réponse impulsionnelle.*

Démonstration. Notons $h=(h_n)_{n\in\mathbb{Z}}$ la réponse impulsionnelle d'un filtre défini par une fonction $\mathcal{F}$.
On a :

$$\begin{aligned}
(y_n)_{n\in\mathbb{Z}} &= \mathcal{F}\big((x_n)_{n\in\mathbb{N}}\big)\\
&= \mathcal{F}\big((\sum_{k\in\mathbb{Z}} x_k\delta^0_{n-k})_{n\in\mathbb{N}}\big)\\
&= \sum_{k\in\mathbb{Z}} x_k\mathcal{F}\big((\delta^0_{n-k})_{n\in\mathbb{N}}\big) \qquad (\mathcal{F}\text{ linéaire})\\
&= \sum_{k\in\mathbb{Z}} x_k\mathcal{F}\big(\tau^{-k}(\delta^0_n)_{n\in\mathbb{N}}\big) \qquad (\text{Déf. de }\tau)\\
&= \sum_{k\in\mathbb{Z}} x_k\tau^{-k}\mathcal{F}\big((\delta^0_n)_{n\in\mathbb{N}}\big) \qquad (\mathcal{F}\text{ invariante en temps})\\
&= \sum_{k\in\mathbb{Z}} x_k\tau^{-k}(h_n)_{n\in\mathbb{N}} \qquad (\text{Déf. de }(h_n)_{n\in\mathbb{Z}})\\
&= \sum_{k\in\mathbb{Z}} x_k(h_{n-k})_{n\in\mathbb{N}} \qquad (\text{Déf. de }\tau)\\
&= (\sum_{k\in\mathbb{Z}} x_k h_{n-k})_{n\in\mathbb{N}} \qquad (5.3)
\end{aligned}$$

On voit donc que le calcul de y nécessite seulement la connaissance de la suite $h=(h_n)_{n\in\mathbb{Z}}$. □

La formule (5.3) est une formule de convolution. On voit ainsi une relation algébrique très simple entre l'entrée et la sortie d'un filtre linéaire, invariant en temps :

$$y = h\star x,$$

qui montre que de tels filtres ne font, in fine, que réaliser des produits de convolution.

Causalité

Dans les exemples que nous avons traités jusqu'à présent, l'indice n des signaux numériques représente le temps. Dans ce cadre, il est impossible qu'un filtre élabore une réponse en fonction de données du futur. C'est ce qui motive l'introduction de la définition suivante.

Définition 18. *Un filtre est dit causal si, à n fixé, y_n ne dépend que des valeurs x_k, pour $k \leq n$ et y_k, pour $k \leq n-1$.*

La réponse impulsionnelle h d'un tel filtre est donc nécessairement à support dans $\mathbb{N}$.
On a alors un équivalent du théorème 10.

Théorème 11. *Un filtre linéaire, invariant en temps et causal vérifie :*

$$y_n = \sum_{k \leq n} h_{n-k} x_k = \sum_{k \in \mathbb{N}} h_k x_{n-k},$$

où l'on a noté $h = (h_n)_{n \in \mathbb{N}}$ la réponse impulsionelle du filtre.

Il existe cependant des cas où il est nécessaire de considérer des filtres non causaux. En traitement d'image par exemple, où les signaux – les images donc – sont de dimension 2, les filtres appliquent à chaque pixel de l'image une fonction de l'ensemble des pixels de l'image considérée. La notion de causalité dans ce cas n'a pas vraiment de sens[3].

Réponse impulsionnelle finie (RIF) et infinie (RII)

Les filtres linéaires invariants en temps et causaux peuvent maintenant être divisés en deux catégories.

Définition 19. *Un filtre linéaire, invariant en temps et causal est dit à réponse impulsionnelle finie (en abrégé RIF ou FIR) si $h = (h_n)_{n \in \mathbb{Z}}$ est à support fini. Il est dit à réponse impulsionnelle infinie (en abrégé RII ou IIR) dans le cas contraire.*

Quasiment par définition, les filtres récursifs sont à réponse impulsionnelle infinie.

5.2 Stabilité

On introduit maintenant une notion importante pour la conception des filtres.

5.2.1 Filtres stables

De manière imagée, on parle de *filtre stable* si le filtre n'explose pas pour certaines entrées. La plupart des filtres que l'on considère sont évidemment stables. Les filtres instables donnent cependant parfois lieu à des phénomènes de saturation, qui peuvent être mis à profit dans certaines applications[4].

Définition 20. *Un filtre est dit stable si il laisse stable le sous-espace $\ell^\infty(\mathbb{Z})$.*

Autrement dit, pour toute entrée bornée, la sortie est bornée.

Remarque 10. *Cette définition est valable pour tout filtre, même non-linéaire.*

[3] C'est le cas de le dire.

[4] Par exemple en électronique, où l'utilisation des Amplificateurs Opérationnels en régime saturé permet de concevoir des oscillateurs et donc des horloges.

5.2.2 Caractérisation

Dans le cas des filtres linéaires, invariants en temps, on a une caractérisation simple de la stabilité.

Lemme 10. *Un filtre linéaire, invariant en temps est stable si et seulement si sa réponse impulsionnelle est une suite de* $\ell^1(\mathbb{Z})$.

Démonstration. Condition suffisante : supposons $h = (h_n)_{n\in\mathbb{Z}} \in \ell^1(\mathbb{Z})$ et $x = (x_n)_{n\in\mathbb{Z}} \in \ell^\infty(\mathbb{Z})$. Alors :

$$y_n = (h \star x)_n \leq \sum_{k\in\mathbb{Z}} |h_k|.|x_{n-k}| \leq \|x\|_\infty . \sum_{k\in\mathbb{Z}} |h_k| = \|x\|_\infty \|h\|_1,$$

ce qui montre que y est bornée, donc dans $\ell^\infty(\mathbb{Z})$.
Condition nécessaire : supposons, pour simplifier, que h soit à valeurs réelles.
Posons $x = \big(\text{signe}(h_{-n})\big)_{n\in\mathbb{Z}}$. Puisque le système est stable, $h \star x \in \ell^\infty(\mathbb{Z})$, or :

$$(h \star x)_0 = \sum_{k\in\mathbb{Z}} h_k x_{-k} = \sum_{k\in\mathbb{Z}} |h_k|.$$

On en déduit le résultat. □

Ce lemme n'est qu'une reformulation du fait que le dual de $\ell^1(\mathbb{Z})$ est $\ell^\infty(\mathbb{Z})$.

5.3 Filtres linéaires récursifs causaux

On considère maintenant des filtres linéaires récursifs causaux qui, dans le cadre fixé par la remarque 9, élaborent leur réponse au temps n par une formule du type :

$$y_n = \sum_{k=1}^{N} a_k y_{n-k} + \sum_{k=0}^{M} b_k x_{n-k}. \tag{5.4}$$

L'intérêt de tels filtres est qu'ils sont très faciles à implémenter en pratique.

5.3.1 Transformée en z

Dans cette partie, on définit un concept important pour l'étude des filtres linéaires récursifs causaux.

Définition 21. *Étant donné un signal* $x = (x_n)_{n\in\mathbb{Z}}$, *on appelle transformée en* z *de* x, *la fonction :*

$$X(z) = \sum_{n\in\mathbb{Z}} x_n z^{-n}.$$

Traditionnellement, on utilise des lettres majuscules pour noter la transformée en z.

5.3.2 Fonction de transfert

Puisque cette nouvelle transformée s'applique à tout signal numérique, on peut en particulier l'appliquer à la réponse impulsionnelle. On peut alors établir un lien entre les transformées en z de cette dernière, de l'entrée et de la sortie.

Lemme 11. *Soit* x *et* y *l'entrée et la sortie du système défini par* (5.4) *et de réponse impulsionnelle* h. *On a :*

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}.$$

Démonstration. Commençons par établir la deuxième égalité. On a :

$$\begin{aligned} Y(z) &= \sum_{n\in\mathbb{Z}} y_n z^{-n} \\ &= \sum_{n\in\mathbb{Z}} \Big(\sum_{k=1}^{N} a_k y_{n-k}\Big) z^{-n} + \sum_{n\in\mathbb{Z}} \Big(\sum_{k=0}^{M} b_k x_{n-k}\Big) z^{-n} \\ &= \sum_{k=1}^{N} a_k z^{-k} \Big(\sum_{n\in\mathbb{Z}} y_{n-k} z^{-(n-k)}\Big) + \sum_{k=0}^{M} b_k z^{-k} \Big(\sum_{n\in\mathbb{Z}} x_{n-k} z^{-(n-k)}\Big) \\ &= \sum_{k=1}^{N} a_k z^{-k} Y(z) + \sum_{k=0}^{M} b_k z^{-k} X(z), \end{aligned} \tag{5.5}$$

d'où le résultat.
La première égalité provient quant à elle du fait que la transformée en z, comme le font les différentes transformées de Fourier, transforme le produit de convolution en produit. □

On voit facilement que la fonction de transfert d'un filtre récursif causal est une fraction rationnelle. La réciproque n'est pas vraie. On peut par exemple voir que le filtre défini par la formule :

$$y_n = x_{n+1}$$

à pour fonction de transfert $H(z) = z$.

Définition 22. *On appelle forme normalisée de la fonction de transfert d'un filtre linéaire récursif causal est une fraction rationnelle*

$$H(z) = \frac{\Pi_{k=1}^{M}(1 - z_k z^{-1})}{\Pi_{k=0}^{N}(1 - p_k z^{-1})},$$

où les coefficients z_k, $k = 1...M$ et p_k, $k = 0...M$ sont appelés respectivement zéros et pôles de la fonction transfert.

Dans la définition précédente, on suppose bien entendu que la fraction rationnelle est irréductible, c'est-à-dire qu'aucun facteur n'apparaît à la fois au numérateur et au dénominateur.

5.3.3 Stabilité

La fonction de transfert introduite à la section précédente permet de formuler un nouveau critère de stabilité des filtres linéaires récursifs, causaux.

Lemme 12. *Un filtre linéaire, récursif, causal est stable si et seulement si les pôles de sa fonction de transfert (mise sous forme irréductible) sont situés à l'intérieur du disque unité.*

Démonstration. Par factorisation, on se ramène au cas où la fonction de transfert s'écrit :

$$H(z) = \frac{1}{1 - pz^{-1}}.$$

Le développement en série entière par rapport au paramètre z^{-1} de cette fraction vaut :

$$H(z) = \sum_{k\in\mathbb{N}} p^k z^{-k}.$$

Par identification, on voit que $H(z)$ est la transformée en z de la réponse impulsionnelle h définie par $h_n = p^n$, qui appartient à $\ell^1(\mathbb{Z})$ si et seulement si $|p| < 1$. □

Il existe de nombreux critères permettant de déterminer rapidement par calcul si les racines d'un polynôme donné sont situées à l'intérieur ou à l'extérieur du disque unité. On peut citer par exemple :
- le critère de Schur,
- le critère de Routh,
- le critère de Jary,
- le critère de Nyquist,
- le critère d'Evans,
- le critère du revers,
- le critère du contour de Hall.

On n'aborde ici ni les énoncés ni les démonstrations de la validité de ces critères. Une fois un critère choisi, il est facile de décider si un filtre est stable ou non.

5.4 Réponse fréquentielle

On va maintenant présenter un lien entre transformée en z et série de Fourier. Ce lien avait déjà été entrevu lorsque l'on avait utiliser un argument de convolution dans la preuve du lemme 11

5.4.1 Définition

La *réponse fréquentielle* d'un filtre (linéaire, récursif ou non, causal ou non) donne des informations sur la manière dont le filtre réagit aux excitations périodiques.

Définition 23. *On appelle réponse fréquentielle d'un filtre la fonction :*

$$\begin{array}{ccc} [0,2\pi] & \to & \mathbb{C} \\ \omega & \mapsto & H(\mathrm{e}^{i\omega}). \end{array}$$

Cette fonction est parfois, par abus de notations, simplement notée $H(\omega)$.

On remarque que $H(\mathrm{e}^{i\omega})$ est la série de Fourier dont les coefficients sont ceux de la réponse impulsionnelle. La réponse en fréquence est donc la transformée de Fourier de la série $(h_n)_{n\in\mathbb{Z}}$.

5.4.2 Réponse à phase linéaire

Dans les applications, il est utile que la *phase* du filtre, c'est-à-dire l'argument de la fonction de transfert, soit une fonction affine de ω. Le filtre est alors appelé abusivement *filtre à phase linéaire*. Expliquons rapidement pourquoi.
Notons $\phi(\omega) = \arg\Big(H(\mathrm{e}^{i\omega})\Big)$ la phase d'un filtre donné (de fonction de transfert H) et considérons deux signaux $t \mapsto \mathrm{e}^{i\omega_1 t}$ et $t \mapsto \mathrm{e}^{i\omega_2 t}$. L'effet du temps sur le déphasage entre les deux signaux est d'introduire un déphasage proportionnel à $\omega_2 - \omega_1$. Dans de nombreuses situations, on souhaite qu'un filtre n'affecte pas plus la phase que ne le ferait le temps, c'est-à-dire que le déphasage entre deux signaux qu'il reçoit soit proportionnel à la différence de leurs phases, ce qui conduit à l'équation :

$$\phi(\omega_2) - \phi(\omega_1) = \alpha(\omega_2 - \omega_1),$$

autrement dit, la fonction de transfert du filtre doit être de la forme :

$$H(\mathrm{e}^{i\omega}) = |H(\mathrm{e}^{i\omega})|\mathrm{e}^{i(\alpha\omega+\beta)}.$$

Dans le chapitre suivant, nous verrons comment obtenir en pratique de tels filtres. Dans le cas où le filtre n'est pas à phase linéaire, la déformation qu'il engendre sur le signal d'entrée est appelée *distorsion de phase*. Ce défaut ne doit pas être confondu avec la *distorsion harmonique* qui intervient lorsque le filtre n'est pas linéaire.

5.5 Notes bibliographiques

Pour une introduction plus complète des filtres numériques, on pourra consulter les chapitres 2, 3, 4, 5 et 9 de la référence [3]. Pour un exposé plus court et en français, on pourra se référer à [5].

Chapitre 6

Conception de filtres numériques

On aborde maintenant un point clef de la dernière partie de ce cours, la conception des filtres. On découpe la présentation en deux parties : l'une sur les filtres à réponse impulsionnelle finie et l'autre, plus succincte, sur les filtres à réponse impulsionnelle infinie.

6.1 Remarques préliminaires

Avant de s'attaquer aux algorithme de conception, on présente une méthode d'obtention de filtre à phase linéaire et on explique comment sont habituellement spécifiées les contraintes sur un filtre.

6.1.1 Problème de la phase linéaire sur un exemple

On souhaite trouver des conditions suffisantes sur les coefficients d'un filtre pour que sa phase soit linéaire. En fait, ceci n'est possible exactement que pour les filtres RIF. Avec les filtres RII, on doit seulement se contenter d'une phase approximativement linéaire, engendrant une faible distorsion de phase.

Obtention d'un filtre RIF non causal à phase nulle

Considérons un filtre RIF non causal, de fonction de transfert G, comportant $N = 2p + 1$ coefficients $(g_k)_{k=-p,\dots,p}$. La réponse fréquentielle de ce filtre est donnée par :

$$G(\omega) = \sum_{k=-p}^{p} g_k \mathrm{e}^{ik\omega} = \sum_{k=-p}^{p} g_k \cos(k\omega) - i \sum_{k=-p}^{p} g_k \sin(k\omega).$$

On remarque que la phase de ce filtre est nulle si et seulement si :

$$\forall \omega \in \mathbb{R},\ \sum_{k=-p}^{p} g_k \sin(k\omega) = 0.$$

Cette condition est réalisée si et seulement si la suite $(g_k)_{k=-p,\dots,p}$ est paire (par rapport à k).

Obtention d'un filtre RIF à phase linéaire causal

Le problème du filtre G de la section précédente est qu'il n'est pas causal, c'est-à-dire non réalisable en pratique dans de nombreux exemples. Pour le rendre causal, il suffit de le décaler de p pas vers la droite. Ceci conduit à définir le filtre $(h_k)_{k=0,\dots,N}$, par la formule :

$$h_k := g_{k-p},$$

ou encore :

$$H(z) := z^{-p} G(z).$$

On a alors :

$$H(\omega) = \mathrm{e}^{-i\omega p} \underbrace{G(\omega)}_{\text{de phase nulle.}},$$

ce qui donne bien une phase linéaire[1]. On voit ici la nécessité d'un filtre à réponse impulsionnelle finie. C'est en effet parce-qu'il ne comporte qu'un nombre fini de coefficients que l'on peut effectuer le décalage permettant au filtre de devenir causal.

Remarque 11. *En décidant d'avoir une phase égale à $\pm\frac{\pi}{2}$, ce qui convient également[2], on aurait trouvé comme condition que la suite $(g_k)_{k=-p,\dots,p}$ soit impaire.*

Classification des filtres RIF causaux à phase linéaire

Le cas où l'entier N est paire se traite de la même manière. En résumé, on distingue habituellement 4 types de filtres RIF à phase linéaire :

Type	N	symétrie
I	impair	$h_n = h_{N-1-n}$
II	pair	$h_n = h_{N-1-n}$
III	impair	$h_n = -h_{N-1-n}$
IV	pair	$h_n = -h_{N-1-n}$

Dans tous les cas, le filtre introduit un retard de $\frac{N-1}{2}$ et une relation de symétrie entre les coefficients par rapport à $k = \frac{N-1}{2}$. Le retard dépend donc du nombre de coefficients du filtre. Comme on le verra dans la suite, pour avoir des filtres précis, il faut avoir beaucoup de coefficients et le prix à payer est donc un retard en réponse important, dû d'une part au temps de calcul et au décalage introduit par le filtre. Les figures 6.1–6.4 donnent des exemples de réponses impulsionnelles des quatre types de filtres.

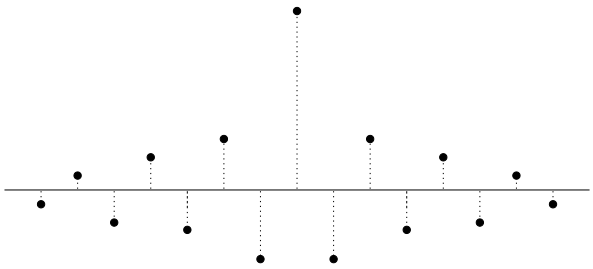

FIG. 6.1 – Réponse impulsionnelle d'un filtre de Type I

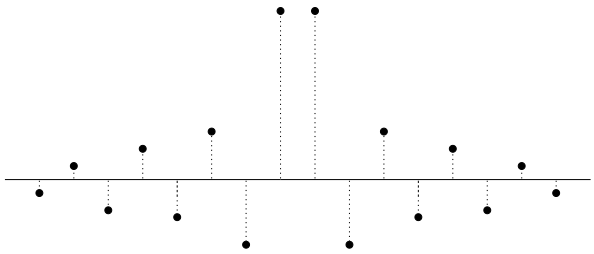

FIG. 6.2 – Réponse impulsionnelle d'un filtre de Type II

6.1.2 Filtres idéaux et gabarits

On raisonne sur la plage de fréquence $[0, \frac{f_e}{2}]$, où f_e est la fréquence d'échantillonnage, ce qui correspond à la plage de pulsation $[0, \pi f_e]$. Ce qui se cache derrière cette règle est bien entendu le théorème de Shannon-Nyquist. On suppose en effet que l'on est dans son cadre d'application, c'est-à-dire que son contenu fréquentiel est inclus dans $[0, \frac{f_e}{2}]$. De plus, puisqu'on se place après la phase d'échantillonage, le temps n'apparaît plus explicitement[3] et l'on peut considérer $f_e = 1$ par commodité.
Dans la pratique, trois filtres sont généralement considérés :

[1] dans ce cas, la phase est même linéaire au sens propre.
[2] dans ce cas, la phase serait affine.
[3] il est caché dans le pas de temps entre x_n et x_{n+1}, soit $\frac{1}{f_e}$ et doit être transmis d'une manière ou d'une autre au cours de la conversion numérique-analogique.

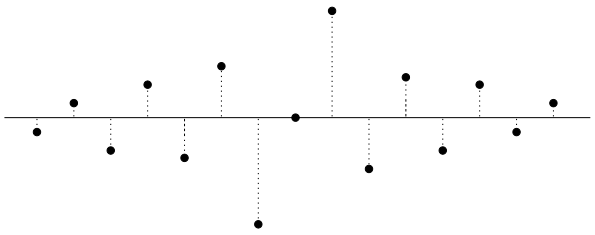

FIG. 6.3 – Réponse impulsionnelle d'un filtre de Type III

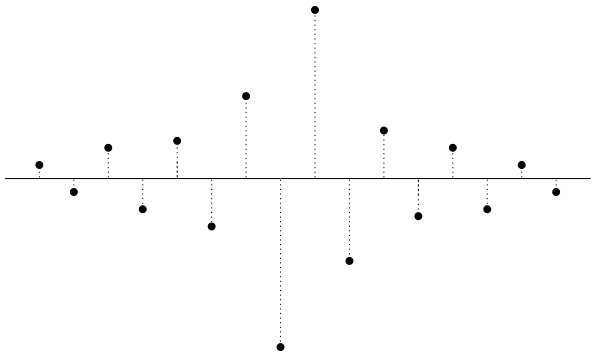

FIG. 6.4 – Réponse impulsionnelle d'un filtre de Type IV

- les filtres passe-haut,
- les filtres passe-bas,
- les filtres passe-bande.

Ces filtres permettent de sélectionner dans un signal respectivement les hautes fréquences, les basses fréquences, ou bien une bande spécifique de fréquence. On appelle généralement *fréquence de coupure* toute fréquence autour de laquelle le filtre change de comportement, c'est-à-dire par exemple une fréquence à partir de laquelle il commence à occulter les composantes de Fourier. La qualité d'un filtre se mesure bien souvent à ses qualités de coupure : on souhaite en pratique avoir une rupture nette du comportement, idéalement passer de 1 à 0 au passage de la fréquence de coupure d'un filtre passe-bas par exemple.
On appelle *gabarit* l'ensemble des contraintes spécifiées sur le filtre. Le terme de gabarit est parfois employé au sens de filtre idéal, c'est-à-dire un filtre à réponse en fréquence constante par morceaux.
En continuant de considérer un filtre passe-bas comme exemple, on spécifiera f_c, la fréquence de coupure, au-delà de laquelle les fréquences ne devront plus être transmises, Δf la zone de coupure dans laquelle doit être effectuée la transition, Δe_1 et Δe_2 les tolérances accordées respectivement autour de 1 et de 0 dans les deux zones de l'intervalle $[0, 1/2]$.
Il faut enfin signaler que l'amplitude de la réponse fréquentielle se mesure parfois en décibels (unité notée dB), c'est-à-dire :

$$amplitude(\omega) = 10 \log_{10} |H(\omega)|^2 = 20 \log_{10} |H(\omega)|.$$

6.2 Conception de filtres RIF

Étant donné un gabarit, on présente deux méthodes de calcul des coefficients d'un filtre RIF respectant les caractéristiques spécifiées dans le gabarit.

6.2.1 Méthode de la fenêtre

Cette méthode est connue pour sa simplicité.

Spécifications

On commence par spécifier une réponse fréquentielle idéale $H^\star(\omega)$, dont on détermine la transformée de Fourier inverse – c'est-à-dire la série de Fourier inverse, puisqu'on est dans le cas périodique – que l'on note $h^\star = (h^\star_k)_{k\in\mathbb{Z}}$ et qui est donnée par les formules :

$$H^\star(\omega) = \sum_{n=\mathbb{Z}} h^\star_n \mathrm{e}^{i\omega n},$$

$$h^\star_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} H^\star(\omega) \mathrm{e}^{i\omega n} \mathrm{d}\omega.$$

Dans la pratique, $H^\star(\omega)$ est souvent de la forme :

$$H^\star(\omega) = \mathbb{1}_{[\omega_1,\omega_2]}(\omega)\mathrm{e}^{-i\omega\frac{M-1}{2}}, \tag{6.1}$$

avec $0 \leq \omega_1 \leq \omega_2 \leq \pi$, et M un entier positif. Le terme exponentiel est introduit car on sait par avance que l'on va tronquer la suite $h^\star$ à partir du coefficient $M-1$.

Troncature

Comme annoncé, et comme c'était le cas dans l'exemple traité à la section 6.1.1, on tronque ensuite la suite de coefficients $(h_k^\star)_{k\in\mathbb{Z}}$ obtenue au rang $M-1$. Ceci revient en fait à multiplier la suite $h^\star$ par la fenêtre rectangulaire $g = (g_k)_{k\in\mathbb{Z}}$:

$$g_n = \begin{cases} 0 & n = 0, ..., M-1 \\ 1 & sinon \end{cases},$$

ce qui correspond à une fonction de transfert définie par la formule :

$$H(\omega) := H^\star(\omega) \star G(\omega),$$

où G est la fonction de transfert du filtre associée à la suite w, et $G(\omega)$ sa réponse en fréquence.
Dans le cas où $H^\star(\omega)$ est de la forme (6.1), on vérifie aisément que la phase du filtre $H^\star$ est linéaire. Cette propriété est bien conservée après multiplication (resp. convolution, selon le domaine où que l'on considère) g (resp. G), puisque la symétrie des coefficients de $h^\star$ est conservée.

Améliorations

Malheureusement, le *phénomène de Gibbs* garantit que le filtre H ne pourra pas tendre assymptotiquement vers $H^\star$ lorsque cette dernière fonction est de la forme (6.1). Pour palier ce problème, on choisit souvent des fenêtres plus régulières, dont les coefficients sont donnés dans des tableaux de référence. On peut par exemple citer les fenêtres de Hamming, Hanning, Barlett...Celles-ci permettent à $H^\star \star W$ de ressembler plus à $H^\star$ que ne le fait la simple fenêtre rectangulaire.

6.2.2 Méthode de l'approximation optimale de Tchebytchev

Développée plus récemment, la méthode de l'approximation optimale de Tchebytchev[4] aborde le problème de la conception de filtre sous l'angle de l'optimisation.

Formulation du problème

On remarque que les réponses fréquentielles obtenues à partir des 4 types de filtres à phase linéaire présentés à la section 6.1.1 peuvent s'écrire sous la forme :

$$H(\omega) = \pm|H(\omega)|\mathrm{e}^{i\phi(\omega)},$$

où on continue de noter abusivement $H(\omega)$ la fonction $H(\mathrm{e}^{i\omega})$ et où $\phi(\omega) = \alpha\omega + \beta$, avec $\beta = 0$ ou $\frac{\pi}{2}$ et $\alpha = \frac{N-1}{2}$.
Au début des années 70, McClellan[5] et son thésard Parks[6] observent que $|H(\omega)|$ peut s'écrire sous la forme :

$$|H(\omega)| = Q(\mathrm{e}^{i\omega})P(\mathrm{e}^{i\omega}),$$

où P est un polynôme ne contenant que des termes en $\cos(n\omega)$ et Q un polynôme de degré au plus 1. Les différents cas sont résumés dans le tableau suivant :

[4] Pafnouti Lvovitch Tchebychev (4 mai 1821, Okatovo - 26 novembre 1894, Saint-Pétersbourg) était un mathématicien russe. Il est connu pour ses travaux dans le domaine des probabilités et des statistiques. Après lui Liapounov et Markov, ses élèves, continueront son Õ153uvre et cette tradition russe conduit à Kolmogorov, fondateur des probabilités contemporaines.

[5] James H. McClellan est un enseignant chercheur en traitement du signal au Georgia Institute of Technology.

[6] Thomas W. Parks, professeur à l'université de Cornell.

Type	$\lvert H(\omega)\rvert$	$P(\mathrm{e}^{i\omega})$	$Q(\mathrm{e}^{i\omega})$
I	$\sum_{n=0}^{\frac{N-1}{2}} a_n \cos(n\omega)$	$\sum_{n=0}^{\frac{N-1}{2}} a_n \cos(n\omega)$	1
II	$\sum_{n=1}^{\frac{N}{2}} b_n \cos((n-\frac{1}{2})\omega)$	$\sum_{n=0}^{\frac{N}{2}-1} b'_n \cos(n\omega)$	$\cos(\frac{\omega}{2})$
III	$\sum_{n=1}^{\frac{N-1}{2}} c_n \sin(n\omega)$	$\sum_{n=0}^{\frac{N-3}{2}} c'_n \cos(n\omega)$	$\sin(\omega)$
IV	$\sum_{n=1}^{\frac{N}{2}} d_n \sin((n-\frac{1}{2})\omega)$	$\sum_{n=0}^{\frac{N}{2}-1} d'_n \cos(n\omega)$	$\sin(\frac{\omega}{2})$

où les formules des coefficients sont données par :

- Type I : pas de changement,
- Type II :

$$\begin{aligned} b'_0 &= \frac{1}{2}b_1 \\ b'_k &= 2b_k - b'_{k-1},\ k = 1, 2, ..., \frac{N}{2} - 2 \\ b'_{\frac{N}{2}-1} &= 2b_{\frac{N}{2}}, \end{aligned}$$

- Type III :

$$\begin{aligned} c'_{\frac{N-3}{2}} &= c_{\frac{N-1}{2}} \\ c'_{\frac{N-5}{2}} &= c_{\frac{N-3}{2}} \\ c'_{k-1} - c'_{k+1} &= 2c_k,\ k = 2, 3, ..., \frac{N-5}{2} \\ c'_0 + \frac{1}{2}c'_2 &= c_1, \end{aligned}$$

- Type IV :

$$\begin{aligned} d'_{\frac{N}{2}-1} &= 2d_{\frac{N}{2}} \\ d'_{k-1} - d'_k &= 2d_k,\ k = 2, 3, ..., \frac{N}{2} - 1 \\ d'_0 - \frac{1}{2}d'_1 &= d_1. \end{aligned}$$

Dans ce cadre, il est possible de reformuler notre le problème d'approximation. En effet, sachant que Q est connu – puisqu'il est fixé en choisissant le type de filtre que l'on souhaite synthétiser – notre inconnue est le polynôme P, ou plutôt ses coefficients, ce qui fait que l'on se ramène au problème de la recherche d'un polynôme de meilleur approximation. Détaillons la démarche.
On se donne donc une réponse fréquentielle $H^\star$ idéale, ainsi qu'une fonction de pondération W permettant de donner plus ou moins d'importance à certaines zones du spectre. On définit alors l'erreur d'approximation par la formule :

$$\begin{aligned} E(\mathrm{e}^{i\omega}) &:= W(\mathrm{e}^{i\omega})[H^\star(\mathrm{e}^{i\omega}) - Q(\mathrm{e}^{i\omega})P(\mathrm{e}^{i\omega})] \\ &= W(\mathrm{e}^{i\omega})Q(\mathrm{e}^{i\omega})[\frac{H^\star(\mathrm{e}^{i\omega})}{Q(\mathrm{e}^{i\omega})} - P(\mathrm{e}^{i\omega})], \end{aligned} \tag{6.2}$$

cette dernière formule étant correctement définie sur $\omega \in [0, \pi]$, sauf éventuellement aux points d'annulation de $\omega \mapsto Q(\mathrm{e}^{i\omega})$, qui sont de toute manière connus. Il est commode de réécrire (6.2) sous la forme :

$$E(\mathrm{e}^{i\omega}) = \widehat{W}(\mathrm{e}^{i\omega})[\widehat{H^\star}(\mathrm{e}^{i\omega}) - P(\mathrm{e}^{i\omega})],$$

où :

$$\widehat{W}(\mathrm{e}^{i\omega}) = W(\mathrm{e}^{i\omega})Q(\mathrm{e}^{i\omega}),$$

et :

$$\widehat{H^\star(\mathrm{e}^{i\omega})} = \frac{H^\star(\mathrm{e}^{i\omega})}{Q(\mathrm{e}^{i\omega})}.$$

Le problème que l'on considère peut dans ce cadre être défini par :

$$\inf_{\text{coeff. de } P} \sup_{\omega\in[0,\pi]} |E(\mathrm{e}^{i\omega})|. \tag{6.3}$$

Caractérisation de l'optimum

Grâce aux travaux de Tchebytchev, on connaît une caractérisation de la solution du problème (6.3). Celle-ci est résumée dans le théorème suivant.

Théorème 12. *(Théorème d'alternance de Tchebytchev) Soit M le degré du polynôme trigonométrique P. Supposons qu'il existe au moins $M+2$ fréquences distinctes $0 \leq \omega_1 < ... < \omega_{M+2} \leq \pi$, telles que :*

1. *La fonction E équioscille sur ces points, c'est-à-dire :*

$$E(\mathrm{e}^{i\omega_1}) = -E(\mathrm{e}^{i\omega_2}) = E(\mathrm{e}^{i\omega_3}) = ...,$$

2. *Le maximum est atteint sur ces points, c'est-à-dire :*

$$\forall k = 1, ..., M+2, \ \max_{\omega\in[0,\pi]} |E(\mathrm{e}^{i\omega})| = |E(\mathrm{e}^{i\omega_k})|,$$

alors, P est l'unique solution du problème (6.3).

La preuve de ce théorème est élémentaire, mais un peu longue et technique. On pourra se reporter au document disponible en ligne "A simple proof of the alternation theorem", par P.P. Vaidyanathan et Q.N. Truong pour une démonstration complète.

Algorithme de Remez

Le théorème d'alternance 12 donne donc un ensemble de conditions suffisantes d'obtention d'une solution. Malheureusement, cet ensemble de conditions ne semble pas déboucher sur une méthode constructive. Pour autant, en 1934, E. Remez[7], a proposé un algorithme reposant sur le théorème d'alternance pour résoudre le problème du polynôme de meilleure approximation.
La méthode consiste à trouver $M+2$ fréquences $0 \leq \omega_1 < ... < \omega_{M+2} \leq \pi$ vérifiant :

$$\widehat{W}(\mathrm{e}^{i\omega_k})[\widehat{H^\star}(\mathrm{e}^{i\omega_k}) - P(\mathrm{e}^{i\omega_k})] = (-1)^k\delta,$$

où δ est la borne supérieure de l'erreur. Ceci peut s'écrire sous la forme matricielle suivante :

$$\begin{pmatrix} 1 & \cos(\omega_1) & \dots & \cos\left(M\omega_1\right) & \frac{1}{\widehat{W}(\mathrm{e}^{i\omega_1})} \\ 1 & \cos(\omega_2) & \dots & \cos\left(M\omega_2\right) & \frac{1}{\widehat{W}(\mathrm{e}^{i\omega_2})} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \cos(\omega_{M+2}) & \dots & \cos\left(M\omega_{M+2}\right) & \frac{1}{\widehat{W}(\mathrm{e}^{i\omega_{M+2}})} \end{pmatrix} . \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_M \\ \delta \end{pmatrix} = \begin{pmatrix} \widehat{H^\star}(\mathrm{e}^{i\omega_1}) \\ \widehat{H^\star}(\mathrm{e}^{i\omega_2}) \\ \vdots \\ \widehat{H^\star}(\mathrm{e}^{i\omega_{M+2}}) \end{pmatrix},$$

[7] Evgeny Yakovlevich Remez (1896 - 1975) est un mathématicien ukrainien surtout connu pour son algorithme qu'il conçut à l'université de Kiev en 1934.

où $(\alpha_n)_{n=0,...,M}$ est la suite des coefficients du polynôme P. L'inversion de cette matrice est en pratique délicate. On préfère donc calculer explicitement la valeur de δ, qui s'obtient par la formule :

$$\delta = \frac{\sum_{n=0}^{M} \lambda_n \widehat{H^\star}(\mathrm{e}^{i\omega_n})}{\sum_{n=0}^{M} \frac{(-1)^n \lambda_n}{\widehat{W}(\mathrm{e}^{i\omega_n})}}, \tag{6.4}$$

avec :

$$\lambda_n = \Pi_{\ell=0,\ \ell\neq n}^{M} \frac{1}{\cos(\omega_n) - \cos(\omega_\ell)}.$$

En se basant sur ces calculs, Remez propose un algorithme pour satisfaire assymptotiquement les conditions 1 et 2 du théorème 12.

Algorithme 2. *(Algorithme de Remez)*

1. *Choisir une estimation initiale de $M+2$ fréquences $0 \leq \omega_1 < ... < \omega_{M+2} \leq \pi$.*
2. *Calculer δ par (6.4).*
3. *Calculer l'interpolée de Lagrange de degré M prenant la valeur $(-1)^n\delta$ aux points $\omega_1 < ... < \omega_n < ... < \omega_{M+1}$.*
4. *Sur un ensemble suffisamment dense (contenant par exemple ω_{M+2}) dans $[0,\pi]$, chercher les maxima de l'interpolée. On choisit alors parmi ces maxima $M+2$ fréquences respectant l'alternance (ce qui est possible, d'après le théorème des valeurs intermédiaires) et on s'arrête lorsque ces maxima sont tous égaux (en valeur absolue et à une tolérance prédéfinie près).*

Cet algorithme est très empirique et les preuves théoriques de sa convergence sont assez techniques, si bien que nous ne la détaillerons pas ici. Il est cependant largement utilisé et est par exemple implémenté dans Matlab.
De même, le choix de M à fait lui aussi l'objet de nombreux articles, où des formules également empiriques sont données pour fixer a priori ce paramètre.

6.3 Conception de filtres RII

Les méthodes de conceptions de filtres à réponses impulsionnelles infinies reposent sur des idées complètement différentes des précédentes. Elles se basent toutes sur des filtres analogiques de référence, qu'elles transforment en filtres numériques. On se contente dans cette partie de n'expliquer qu'une des méthodes parmi les plus courantes, la *méthode de la transformation bilinéaire.*

6.3.1 Quelques filtres analogiques célèbres

En électronique non-numérique (qui est en voie de disparition), les filtres utilisées sont caractérisés par des équations différentielles que l'on traite en *régime permanent*, c'est-à-dire qu'on ne considère que les solutions en temps long de ces équations avec second membre de type $\mathrm{e}^{i\Omega t}$. Ces équations étant linéaires, on sait que dans le cas de systèmes stables, la solution au bout d'un temps suffisamment long est elle proportionnelle à $\mathrm{e}^{i\Omega t}$. Le coefficient de proportionnalité est la fonction de transfert du système analogique.
Il existe un certain nombre de fonction de transfert de référence. Par exemple, on appelle *filtre de Butterworth* un filtre dont la fonction de transfert vérifie :

$$|H(\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}},$$

où Ω_c et N sont deux paramètres du filtre appelés respectivement *fréquence de coupure à -3dB* et *ordre* du filtre. Ce filtre est un passe-bas.
Il existe d'autres filtres, également passe-bas, caractérisés de manière analogue au filtre de Butterworth : filtres de Tchebytchev, filtre elliptique, filtre de Bessel... Ils se caractérisent par leur qualité d'atténuation dans les hautes fréquences et par la complexité de leurs mises en oeuvre.

6.3.2 Méthode de la transformation bilinéaire

Ayant à notre disposition une formule caractérisant un filtre analogique, la méthode de la transformation bilinéaire propose une démarche simple pour en déduire un filtre numérique.

Exemple

Considérons par un exemple le filtre passe-bas très simple associé à l'équation différentielle :

$$\frac{dy}{dt} + ay = bx.$$

La fonction de transfert de ce filtre est donnée par :

$$H(\Omega) = \frac{b}{a + i\Omega}, \tag{6.5}$$

ce qui en fait un filtre (proportionnel à un filtre) de Butterworth. On a d'autre part en toute généralité :

$$y(t) = \int_{t_0}^{t} y'(s)\mathrm{d}s + y(t_0).$$

Fixons maintenant un pas d'échantillonnage Δt et un entier n. On vérifie que :

$$y(nT) = \frac{T}{2}\Big(y'\big(nT\big) + y'\big((n-1)T\big)\Big) + y\big((n-1)T\big).$$

Or $y'(nT) = -ay(nT) + bx(nT)$, ce qui conduit à :

$$\Big(1 + \frac{a\Delta t}{2}\Big)y_n - \Big(1 - \frac{a\Delta t}{2}\Big)y_{n-1} = \frac{b\Delta t}{2}(x_n + x_{n-1}).$$

Le filtre numérique associé à cette formule de récurrence à alors pour fonction de transfert :

$$H(z) = \frac{b}{a + \frac{2}{\Delta t}\Big(\frac{1-z^{-1}}{1+z^{-1}}\Big)}. \tag{6.6}$$

La correspondance entre les fonctions de transfert analogique et numérique s'effectue donc par la formule :

$$\Omega = \frac{2}{\Delta t}\Big(\frac{1 - z^{-1}}{1 + z^{-1}}\Big). \tag{6.7}$$

Cette transformation, bien que pas du tout bilinéaire au sens mathématique du terme, est appelée bilinéaire dans la communauté du traitement numérique du signal[8].

Généralisation

La méthode de la transformation bilinéaire consiste alors à appliquer (6.7) à une fonction de transfert analogique pour en obtenir une version numérique. On peut vérifier qu'en posant $z = \mathrm{e}^{i\omega}$ on obtient une autre version de (6.7) :

$$\Omega = \frac{2}{\Delta t}\tan\frac{\omega}{2}$$

soit encore :

$$\omega = 2\arctan\frac{\Omega\Delta t}{2},$$

montre qu'on ne fait en fait que transformer $\mathbb{R}$ en un intervalle de longueur π correspondant parfaitement au paradigme des signaux numériques.

[8] En maths, on appelle plutôt cette transformation une *homographie*.

6.4 Conclusion

Le choix entre filtres RIF et filtres RII s'effectue suivant plusieurs critères. Résumons les qualités et inconvénients propres des deux approches :

- intérêts/défauts des filtres RIF : ils permettent d'obtenir des réponses en fréquences à phase linéaire, c'est-à-dire sans distorsion de phase, ce qui pour certaines applications est une caractéristique cruciale. En contre partie, il nécessite pour avoir des réponses en fréquence très spécifiques beaucoup de coefficients et donc requièrent un coût de calcul ainsi qu'un retard (le décalage présenté à la section 6.2.1) important dans l'élaboration de la réponse.
- intérêts des filtres RII/défauts : ils engendrent des structures légères de calcul car nécessite peu de coefficients pour l'élaboration de la réponse. Par contre ils n'assurent pas exactement la linéarité de la phase de la réponse en fréquence. Un dernier problème posé par les filtres RII, est que leur stabilité peut être affectée par la quantification des coefficients du filtre : les valeurs des coefficients calculées lors de la conception d'un filtre sont nécessairement différentes de celles du filtre qui est réalisé en pratique. Ceci peut avoir pour effet de transformer un filtre stable en un filtre instable en faisant passer un pôle à l'extérieur du cercle unité.

Les deux approches sont donc complémentaires.

6.5 Notes bibliographiques

Une présentation simple et claire de la méthode de la fenêtre est faite dans la référence [5]. Toutes les informations concernant la méthode de l'approximation optimale de Tchebytchev ont été obtenues sur Internet. On rappelle qu'un preuve du théorème d'alternance est donnée dans "A simple proof of the alternation theorem", par P.P. Vaidyanathan et Q.N. Truong et également disponible en ligne.
Pour une présentation moins mathématique de l'ensemble des méthodes de ce chapitre, on pourra se référer au chapitre 10 de la référence [3].

Chapitre 7

Codage en sous-bandes

Dans ce dernier chapitre, on introduit une méthode de compression avec perte, le *codage en sous-bandes*, aussi appelée *filtrage multicadence*. Cette technique permet de décomposer un signal suivant différentes composantes fréquentielles. La compression consiste alors à allouer les bits de codage lors de la numérisations en fonction de l'importance que l'on accorde à chacune des composantes fréquentielle.

7.1 Décomposition en sous-bandes et reconstruction exacte

Dans cette partie, on présente la méthode du codage en sous-bandes et obtient des conditions suffisantes permettant d'assurer une restitution exacte du signal en sortie de canal.

7.1.1 Principe de la méthode

Si elle repose surtout sur une étape de décomposition, la méthode explicite également comment reconstruire le signal. Ces deux aspects sont brièvement présentés dans cette section.

Analyse et décomposition en bandes de fréquences

Considérons un signal analogique stable $x(t)$ par exemple réel que l'on cherche à analyser dans le sens suivant. Pour un entier $N = 2^k$ fixé, on souhaite trouver une suite de signaux $\big(t \mapsto x_i(t)\big)_{i=1\ldots N}$ dont les transformées de Fourier vérifient :

$$\widehat{x_i}(\nu) = \mathbb{1}_{B_i}(\nu)\widehat{x}(\nu),$$

où les plages de fréquences B_i sont données par :

$$B_i = \left[\frac{i-1}{2^k}B, \frac{i}{2^k}B\right].$$

Si la solution de ce problème est en théorie facile à trouver (il suffit de multiplier la transformée de Fourier de x par des indicatrices), on sait qu'en pratique, les filtres ne réalisent jamais un "fenêtrage" parfait. On utilise en pratique des filtres dont les réponses en fréquence approchent les indicatrices $\mathbb{1}_{B_i}$.
On appelle *analyse* cette étape de décomposition.

Synthèse et reconstruction

Après avoir ainsi décomposer le signal, on souhaite pouvoir le reconstruire parfaitement : en effet, si on oublie l'objectif de compression avec perte, il est naturel de chercher à pouvoir retrouver exactement le signal à partir de ses composantes. Celle-ci consiste en premier lieu à une phase de filtrage, car comme on va le voir dans la suite le processus subit par les signaux après la décomposition ajoute des composantes artificielles. Ensuite, suffit d'additionner les différentes composantes pour générer le signal de sortie. On appelle *synthèse* l'étape de reconstruction du signal.

7.1.2 Algorithme de base

On s'intéresse maintenant à la décomposition d'un signal sur 2 bandes. En effet, une fois cette étape comprise, il suffira de la composer k fois par elle-même pour obtenir la décomposition en N bandes recherchée.
Supposons que le contenu fréquentiel du signal d'entrée x, c'est-à-dire le support de $\widehat{x}$, soit contenu dans un intervalle $[-B, B]$. D'après le théorème de Shannon-Nyquist, on peut donc également supposer que ce signal a été échantillonné à une fréquence $2B$ et on note toujours x ce signal. Celui-ci est maintenant envoyé dans deux filtres passe-bas et passe-haut de fonctions de transfert respectives $H_0(z)$ et $H_1(z)$. On note $\widetilde{x}_0$ et $\widetilde{x}_1$ les deux signaux correspondant. Leurs transformées en z vérifient donc :

$$\begin{aligned} \widetilde{X}_0(z) &= H_0(z)X(z) \\ \widetilde{X}_1(z) &= H_1(z)X(z). \end{aligned}$$

Où $X(z)$ désigne la transformée en z du signal initial x. Ce faisant, on double le nombre de bits considérés, puisqu'on a maintenant deux signaux échantillonnés à la même fréquence que le signal initial. On peut corriger ce problème[1] de la manière suivante. En supposant que ces deux filtres soient idéaux, le contenu fréquentiel deux signaux est maintenant contenu dans des intervalles de tailles deux fois plus petites, c'est-à-dire B. On peut donc se permettre de ne les échantillonner qu'à une fréquence B. Ceci revient exactement à ne considérer qu'un bit sur deux des signaux à la sortie de chacun des deux filtres. Cette étape est parfois appelée *décimation*.
On note x_0 et x_1 les deux signaux obtenus et X_0 et X_1 leurs transformées en z. On a alors les relations :

$$\begin{aligned} X_0(z) &= \frac{1}{2}\left(\widetilde{X}_0(z^{\frac{1}{2}}) + \widetilde{X}_0(-z^{\frac{1}{2}})\right) \\ X_1(z) &= \frac{1}{2}\left(\widetilde{X}_1(z^{\frac{1}{2}}) + \widetilde{X}_1(-z^{\frac{1}{2}})\right) \end{aligned}$$

On a ainsi décomposé en deux notre signal et ce à nombre de bits constant. C'est ici qu'interviennent les différents algorithmes de compression avec perte, appliqués de manière indépendante à x_0 et x_1. Ceux-ci peuvent par exemple re-quantifier plus grossièrement l'un des deux, être eux-même suivis de compression sans perte, etc... Ce sont les produits de cette étape qui sont envoyés dans le canal. Puisqu'avant compression, la somme totale des bits était égale à celle du signal initial x, on fait transiter dans le canal un nombre de bit inférieur au nombre initial, ce qui correspond au but recherché. En sortie de canal, la première couche de traitement est constituée des différents décodeurs correspondants aux différents algorithmes de compressions utilisés[2].
Notre but dans ce chapitre n'est pas d'étudier précisément un algorithme particulier de compression avec perte (on peut considérer que cela déjà été traité au chapitre 2), mais de nous concentrer sur la méthode générale du codage en sous-bande. De manière à obtenir un critère de reconstruction parfaite du codeur en sous-bande seul, on suppose donc qu'on obtient après cette dernière étape exactement x_0 et x_1.
La reconstruction se fait alors en trois étapes. On commence par sur-échantillonner les deux signaux. Concrètement, cela revient ajouter un zéro entre chacun des bits de x_0 et x_1. On note $\check{x}_0$ et $\check{x}_1$ les deux signaux obtenus. Leurs transformées en z vérifient donc :

$$\begin{aligned} \check{X}_0(z) &= X_0(z^2) \\ \check{X}_1(z) &= X_1(z^2). \end{aligned} \tag{7.1}$$

Après cette étape, un filtrage est nécessaire, car cette dernière opération à fait apparaître dans le spectre une partie artificielle par périodicité. On ajoute donc de nouveau un filtrage des deux signaux par deux filtres, passe-bas et passe-haut de fonctions de transfert respectives $F_0(z)$ et $F_1(z)$.
Enfin, on ajoute les deux signaux obtenus pour reconstruire le signal initial. La structure ainsi obtenue est généralement appelée *banc de codage en sous-bande*. La figure 7.1 schématise ce banc de codage et le devenir du spectre du signal à travers le banc est représenté sur la figure 7.2.

[1] qui est un réel problème puisqu'on a pour but de décomposer le signal en 2^k signaux.
[2] On passe ici sous silence le codage canal dont il a été question au chapitre 3, qui intervient en tout dernier lieu avant l'envoie du signal dans le canal et dont le décodeur intervient par conséquent en premier en sortie de canal.

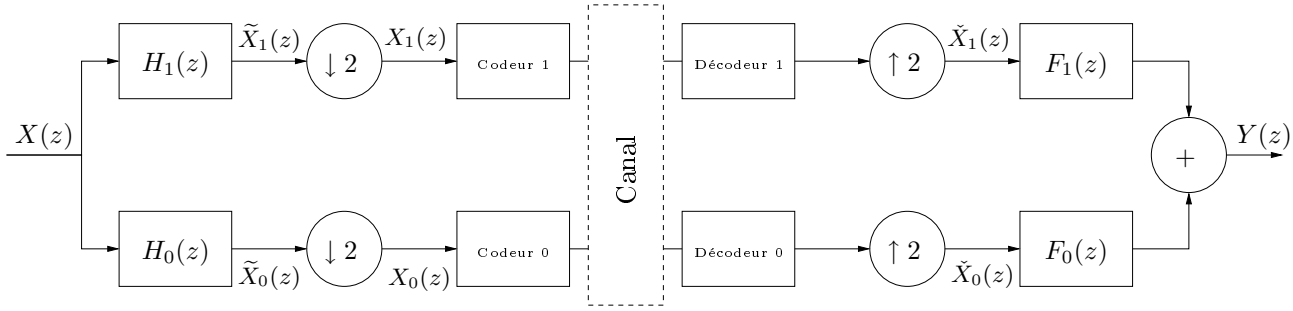

FIG. 7.1 – Banc de codage en sous-bandes.

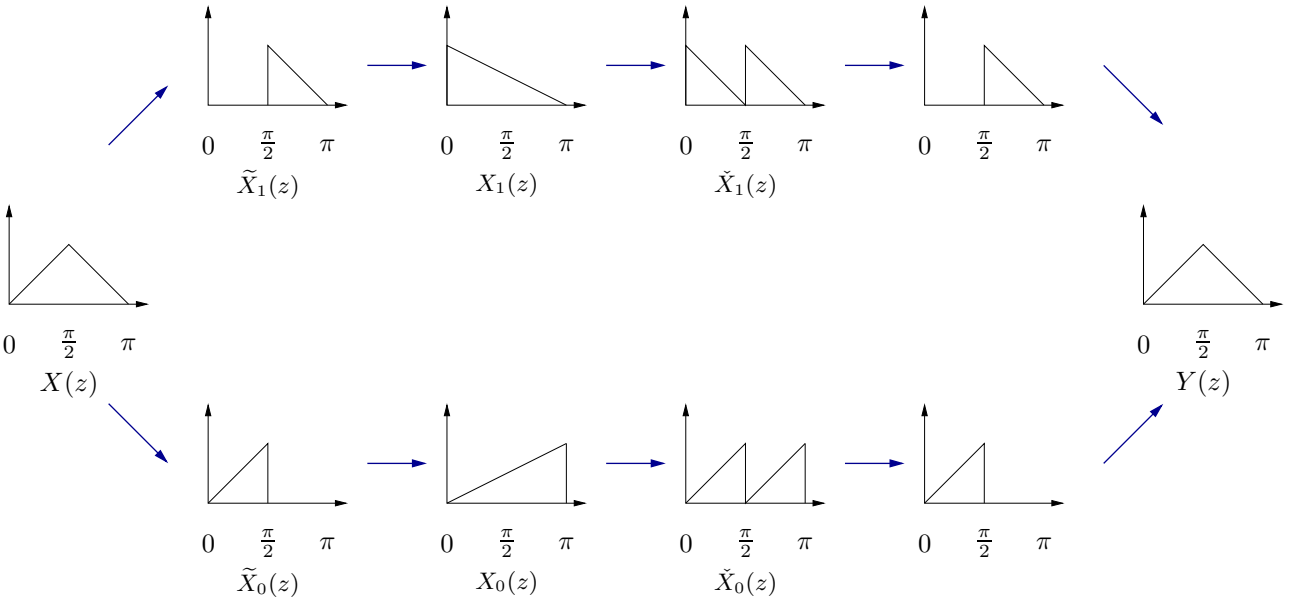

FIG. 7.2 – Représentation schématique des effets du banc sur la réponse fréquentielle du signal.

En notant y et $Y(z)$ le signal de sortie et sa transformée en z, le bilan de toutes ces manipulations est résumé par la formule :

$$\begin{aligned} Y(z) &= \frac{1}{2}\left(H_0(z)F_0(z) + H_1(z)F_1(z)\right)X(z) \\ &\quad + \frac{1}{2}\left(H_0(-z)F_0(z) + H_1(-z)F_1(z)\right)X(-z). \end{aligned}$$

On aura donc une reconstruction parfaite, avec un décalage n_0 si et seulement si :

$$H_0(z)F_0(z) + H_1(z)F_1(z) = 2z^{-n_0} \tag{7.2}$$

$$H_0(-z)F_0(z) + H_1(-z)F_1(z) = 0 \tag{7.3}$$

La condition (7.3) est appelée *condition de non-repliement*, en anglais *antialiasing condition*, car elle permet d'éviter l'apparition de fréquences parasites de la même nature que celles que l'on avait rencontrées lorsque les conditions du théorème de Shannon-Nyquist n'étaient pas satisfaites.

7.2 Filtres utilisés dans le codage en sous-bandes

On s'intéresse maintenant plus particulièrement aux filtres intervenants dans le banc de codage en sous-bandes présenté précédemment.

7.2.1 Filtres en quadrature (QMF)

La condition (7.3) est automatiquement vérifiée si on impose les relations :

$$F_0(z) = H_1(-z),\ F_1(z) = -H_0(-z).$$

Il reste à résoudre (7.2). Une solution consiste à imposer une relation supplémentaire. L'approche dite *QMF*, qui signifie *Quadrature Mirror Filters* suggère ainsi de choisir :

$$H_1(z) = H_0(-z).$$

Le terme mirror vient de cette relation, car celle-ci introduit une relation de symétrie autour de la fréquence $\frac{\pi}{2}$ entre les réponses fréquentielles des filtres H_0 et H_1.
Il ne reste alors plus qu'une inconnue H_0 et l'équation déduite de (7.2) qu'elle doit vérifier est :

$$H_0^2(z) - H_0^2(-z) = 2z^{-n_0}. \tag{7.4}$$

Il est facile de voir que si on choisit pour H_0 un filtre FIR à phase linéaire, alors le terme de gauche de (7.4) sera aussi à phase linéaire. Par conséquent, l'équation (7.4) sera vérifiée pour ce qui concerne les arguments. Pour autant, on peut montre que cette équation n'admet de solutions FIR que pour $n_0 = 0$ ou $n_0 = 1$. Dans ce dernier cas, une solution triviale est donnée par le *filtre de Haar* :

$$H_0(z) = \frac{1}{\sqrt{2}}(1 + z),$$

qui, n'ayant pas une bonne qualité de coupure, n'a malheureusement pas de réelle utilité pratique.
On se tourne donc vers des filtres à phase non-linéaire. On montre que H_0 doit nécessairement être de la forme :

$$H_0(z) = \alpha z^{-2k} + \beta z^{-(2k'+1)},$$

qui n'est pas à phase linéaire. Dans ce cas, on a :

$$T(z) = 2\alpha\beta z^{-2k+2k'+1},$$

qui donne bien le résultat recherché. Mais là encore, il faut noter que les filtres obtenus H_0 et ses filtres dérivés H_1, F_0 et F_1 ne présentent pas de bonnes qualités de coupure, outre le fait qu'ils sont à phase non-linéaire. Ceci complique le travail de compression avec perte, puisque les produits du filtrage ne sont pas très "purs".

7.2.2 Filtres à puissance symétrique

On continue de considérer les filtres vérifiant les relations :

$$F_0(z) = H_1(-z),\ F_1(z) = -H_0(-z),$$

et assurant ainsi la condition (7.3).
Une approche proposée indépendamment par Smith et Barnwell en 1984 d'une part et par Mintzer en 1985 consiste à imposer :

$$H_1(z) = z^{-n_0} H_0(-z^{-1}),$$

avec n_0 impair. Les filtres obtenus sont appelés *filtres à puissances symétriques.*
De la sorte, la condition (7.3) s'écrit :

$$Q(z) = H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = constante.$$

Après un raisonnement d'algèbre élémentaire mais technique, on peut montrer que les solutions de cette équation sont de la forme :

$$H_0(\omega) = \cos^p(\frac{\omega}{2})\sqrt{P\Big(\sin^2(\frac{\omega}{2})\Big)},$$

avec P de la forme :

$$P(X) = (1+X)^{p-1} + X^p R(\frac{1}{2} - X),$$

où R soumis aux seules contraintes d'être pair et tel que P soir positif. On se référera au chapitre B4-2 de [1] pour une preuve complète de ce résultat.

7.2.3 Conditions générales d'orthogonalité

On résume finalement la méthode générale de conception des filtres intervenant dans les bancs de codage en sous-bandes.
Si on note $P(z) = H_0(z)F_0(z)$, on voit que la condition de non-repliement (7.2) impose :

$$H_1(z)F_1(z) = -P(-z),$$

et la condition (7.3) s'écrit :

$$P(z) - P(-z) = 2z^{-n_0}.$$

Cette équation ne peut être vérifiée que si P est de la forme :

$$P(z) = p_{n_0} z^{-n_0} + \sum_{n=0}^{p} p_{2n} z^{-2n}. \tag{7.5}$$

La méthode consiste donc à :

1. choisir un P satisfaisant (7.5),
2. factoriser P en deux filtres H_0 et F_0.

7.3 Notes bibliographiques

Pour plus d'informations sur les aspects mathématiques du codage en sous-bande, en particulier sur les conditions de reconstruction parfaite, on pourra consulter la référence [1]. Pour une présentation plus succincte, mais avec des applications à la compression avec perte, on se reportera à la référence [2].

Chapitre 8

Exercices

8.1 Chapitre 1 – Formule de Shannon-Nyquist et échantillonnage

Exercice 1. Soit s un signal stable, $0 < T < +\infty$ et $\phi = \sum_{n\in\mathbb{R}} s(t+nT)$. Montrer que :

1. ϕ est définie p.p,
2. ϕ est localement stable, $T-$périodique,
3. le $n-$ième coefficient de Fourier de s vaut $\frac{1}{T}\hat{s}(\frac{n}{T})$

Exercice 2. (Identités approchées) On dit que la famille $(h_r)_{r\in[0,1]}$ est une identité approchée si :
- $\int_{\mathbb{R}} h_r = 1$,
- $\forall r \in [0,1]$, $h_r \geq 0$,
- $\forall a > 0$, $lim_{r\to 0}\int_{-a}^{a} h_r(u)du = 1$,

Soit s un signal stable. Montrer que $lim_{r\to 0}\int_{\mathbb{R}} |s \star h_r(t) - s(t)|dt = 0$.

Exercice 3. (Dérivation de convolutions) Soit a de classe $\mathcal{C}^1$, stable de dérivée a' stable et b stable. Montrer l'existence et calculer la dérivée de la fonction $a \star b$.

Exercice 4. Soit a et b deux signaux stable. Montrer que $\|a \star b\|_{L^1} \leq \|a\|_{L^1}\|b\|_{L^1}$
On suppose maintenant $a \in L^1$ et $b \in L^2$. Montrer que :

1. $a \star b$ est définie p.p.
2. $a \star b \in L^2$ et $\|a \star b\|_{L^2} \leq \|a\|_{L^1}\|b\|_{L^2}$.

Exercice 5. (Uniforme continuité de la transformée de Fourier) Montrer que la transformée de Fourier d'un signal stable est uniformément continue.

Exercice 6. (Phénomène de Gibbs) Soit f 2π-périodique définie par : $f(0) = 0$, $f(x) = -\frac{\pi}{2} - \frac{x}{2}$ sur $[-\pi, 0[$ et $f(x) = \frac{\pi}{2} - \frac{x}{2}$ sur $[0, \pi[$.

1. Montrer que $S_n(f)(x) = \sum_{k=1}^{n} \frac{\sin(nx)}{n}$.
2. Montrer que $S_n(f)(x) = \int_0^x \frac{\sin\left((n+1/2)t\right)}{2\sin(t/2)}dt - \frac{x}{2}$.
3. Montrer que :

$$\int_0^x \frac{\sin\left((n+1/2)t\right)}{2\sin(t/2)}dt = \int_0^x \frac{\sin(nt)}{t}dt + \int_0^x \sin(nt)(\frac{\cos(t/2)}{2\sin(t/2)} - \frac{1}{t})dt + \frac{1}{2}\int_0^x \cos(nt)dt.$$

4. En déduire : $\exists A > 0 / \forall n \in \mathbb{N} \qquad S_n(f)(\frac{\pi}{n}) \geq A$.

Exercice 7. Soit $(c_k)_{k\in\mathbb{Z}}$ telle que $S_n(t) = \sum_{k=-n}^{n} c_k e^{ikt}$ converge dans L^1_{loc} vers une fonction f. Montrer que $c_k = \hat{f}_k$.

Exercice 8. Soit f C^1 2π-périodique et $\alpha \in \mathbb{R} - \mathbb{Q}$. Montrer que :

$$\lim_{n\to+\infty} \frac{f(x) + f(x+\alpha) + ... + f(x+(n-1)\alpha)}{n} = \int_0^{2\pi} f(t)dt.$$

Peut-on affaiblir les hypothèses sur f ?

Exercice 9. (Convergence uniforme des sommes de Féjer) Soit f $2-\pi$ périodique et continue. Montrer que :

$$sup_{[-\pi,\pi]}(\sigma_n(f)(x) - f(x)) \to 0,$$

avec :

$$\sigma_n(f)(x) = \frac{1}{n}\sum_{k=1}^{n} S_k(f)(x),$$

où $S_k(f)$ est la somme partielle de rang k de la série de Fourier associée à f.

Exercice 10. (Théorème de Féjer) Soit f $2-\pi$ périodique et localement intégrable.

1. Soit $\delta > 0$. On suppose que :

$$lim_{n\to+\infty}\frac{1}{n}\int_0^{\delta} \sin^2(\frac{1}{2}nu)\frac{\phi(u)}{u^2}du = 0,$$

où $\phi(u) = f(x+u) + f(x-u) - 2A$.
Montrer que :

$$lim_{n\to+\infty}\sigma_n(f)(x) = A.$$

2. (Théorème de Féjer) On fixe $x \in \mathbb{R}$ et on suppose que f admet une limite à gauche et à droite en x. Alors :

$$lim_{n\to+\infty}\sigma_n(f)(x) = \frac{f(x^-) + f(x^+)}{2}.$$

Exercice 11. (Accélération de convergence par sur-échantillonage) On considère un signal s tel que $supp(\hat{s}) \subset [-W, W]$, pour $W > 0$ et $B := (1+\alpha)W$ pour $\alpha > 0$. Soit enfin T tel que $T(\nu) = 1$ sur $[-W, W]$ et $T(\nu) = 0$ sur $]-\infty, -B] \cup [B, +\infty[$.

1. Montrer que :

$$\frac{1}{2B}\sum_{j\in\mathbb{Z}} \hat{s}(\nu + 2jB)T(\nu) = \hat{s}(\nu),$$

et en déduire une identité de type Shannon-Nyquist.

2. On choisit :

$$T(\nu) = \frac{\nu + B}{B - W}1_{[-B,-W]} + 1_{[-W,W]} + \frac{-\nu + B}{B - W}1_{[-B,-W]}.$$

Étudier la vitesse de convergence de la série de l'identité de Shannon.

Exercice 12. (Examen de rattrapage 2006) On échantillonne $e_r(t) = \cos(2\pi rt)$ à une fréquence $f = 1/\Delta$.

1. Rappeler la signification de Δ.
2. Montrer que deux signaux e_r et $e_{r'}$ ont le même échantillonnage si $r\Delta - r'\Delta \in \mathbb{Z}$
3. Expliquer pourquoi et dans quel sens le théorème de Nyquist-Shannon et ses hypothèses garantissent l'unicité du r.

8.2 Chapitre 2 – Quantification scalaire des signaux numériques

Exercice 13. (Variance d'un signal uniforme)

1. Montrer que la variance d'un signal aléatoire uniformément distribué sur $[-X_{max}, X_{max}]$ est donnée par :

$$\sigma_x^2 = \frac{(2X_{max})^2}{12}.$$

2. Retrouver la formule du rapport signale bruit du cours.

Exercice 14. (Quantification uniforme optimale) Calculer une condition d'optimalité pour le problème de la minimisation de σ_p^2 dans le cas d'une quantification uniforme d'un signal de densité non-uniforme.

Exercice 15. (Quantification non-uniforme) Montrer qu'un algorithme de quantification non-uniforme est optimal si :

$$y_i = \frac{\int_{b_{j-1}}^{b_j} x f_X(x) dx}{\int_{b_{j-1}}^{b_j} f_X(x) dx}$$

$$b_j = \frac{y_{j+1} + y_j}{2}.$$

8.3 Chapitre 3 – Codage sans perte de l'information

Exercice 16. (Inégalité de Kraft) On souhaite coder un ensemble de mots $(x_i)_{1 \leq i \leq m}$.

1. Montrer le théorème suivant :
   Théorème (Inégalité de Kraft) : Pour tout code irréductible composé à partir d'un alphabet de taille q, les longueurs $(n_i)_{1 \leq i \leq m}$ des mots de codage vérifient :

$$\sum_{1 \leq i \leq m} q^{-n_i} \leq 1. \quad (K)$$

2. Montrer la réciproque de ce théorème :
   Théorème (réciproque de l'Inégalité de Kraft) : Étant donné un ensemble d'entiers $(n_i)_{1 \leq i \leq m}$ vérifiant (K), montrer qu'il existe un codage irréductible des mots $(x_i)_{1 \leq i \leq m}$, de telle sorte que la longueur du code de x_i soit n_i.

Exercice 17. (Inégalité de Gibbs) Montrer l'inégalité de Gibbs :

Soit n nombres p_i et q_i tels que :

$$0 < p_i < 1, \qquad 0 < q_i < 1,$$

et

$$\sum_{i=1}^{n} p_i = 1, \qquad \sum_{i=1}^{n} q_i \leq 1.$$

Alors :

$$\sum_{i=1}^{n} p_i \log_2 \left(\frac{q_i}{p_i} \right) \leq 0,$$

avec égalité si et seulement si

$$\forall i,\ 0 \leq i \leq n, \qquad p_i = q_i.$$

Exercice 18. Une source émet des lettres d'un alphabet $\{a_1, a_2, a_3, a_4, a_5\}$ avec les probabilités $P(a_1) = .15$,$P(a_2) = .04$,$P(a_3) = .26$,$P(a_4) = .05$,$P(a_5) = .5$.

1. Calculer l'entropie de la source.
2. Construire un code de Huffman pour cette source.

3. Calculer la longueur moyenne après codage.

Exercice 19. (Propriétés des codes de Huffman) Soit C un code optimal (c'est-à-dire qui minimise $\bar{\ell}$) obtenu à partir d'un alphabet binaire.

1. Montrer que si $p_j > p_k$, $n_j \leq n_k$.
2. Montrer que les deux mots les plus longs de ce code ont les mêmes longueurs.
3. Montrer que les codes de Huffman vérifient ces propriétés et que les deux mots de longueurs maximales diffèrent uniquement par leur dernier bit.

Exercice 20. (Examen 2008) On considère l'alphabet $S = s1, s2, s3, s4, s5, s6, s7, s8$ avec la distribution de fréquences f suivante :

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$0,4$	$0,18$	$0,1$	$0,1$	$0,07$	$0,06$	$0,05$	$0,04$

1. Calculer l'entropie de la source d'information S.
2. Calculez un codage de Huffman de cette source, puis la longueur moyenne de ce codage. Comparez cette longueur moyenne à l'entropie.

Exercice 21. On considère une source binaire sans mémoire U de loi de probabilité : $P(U = 0) = 0,9$ et $P(U = 1) = 0,1$. Le symbole 0 étant beaucoup plus fréquents que le symbole 1, on se propose de coder les séquences issues de la source en tenant compte du nombre de zéros entre deux uns consécutifs. L'opération consiste en deux étapes :

- Première étape
  On compte le nombre de zéros entre deux uns consécutifs. On obtient ainsi un entier que l'on appelle entier intermédiaire.
- Deuxième étape
  On code l'entier intermédiaire en un mot binaire constitué de quatre éléments binaires si l'entier est inférieur ou égal à 7 et on choisit un mot de un élément binaire si l'entier est 8. Si l'entier intermédiaire dépasse 8, on codera les suites de 8 zéros consécutifs par un bit correspondant à l'entier intermédiaire autant de fois que nécessaire. On obtient ainsi la table de correspondance entre les séquences source et les entiers intermédiaires :

séquences source	entiers intermédiaires
1	0
01	1
001	2
0001	3
...	...
...	...
00000001	7
00000000	8

1. Les contraintes imposées permettent-elles de choisir un code uniquement déchiffrable (on ne demande pas d'expliciter un code particulier) ?
2. Calculer le nombre moyen n_1 de bits source par entier intermédiaire.
3. Calculer le nombre moyen n_2 de bits encodés par entier intermédiaire.
4. On considère une séquence de bits source de longueur n avec n très grand. En appliquant la loi faible des grands nombres, exprimer le rapport du nombre de bits utilisés pour coder cette séquence au nombre de bits source (n) en fonction de n_1 et n_2 . Calculer numériquement la valeur de ce rapport. Comparer avec la limite en dessous de laquelle il n'est pas possible de descendre.
5. Effectuer un codage de Huffman de l'extension d'ordre 4 de la source U. Calculer (numériquement) le nombre moyen de bits utilisés pour coder un élément binaire issu de la source U. Comparer avec les résultats du 4.

Exercice 22. (Examen 2007) On considère le message

"ceciestuncodagedehuffman"

(on a supprimé les espaces et la ponctuation pour simplifier la construction).

1. Construisez un codage de Huffman du message (Il y a plusieurs codages de Huffman possibles). On indiquera à chaque noeud de l'arbre le poids de son "sous-arbre".
2. Combien de bits seraient nécessaire pour transmettre le message sans codage ? (Plusieurs réponses possibles, précisez bien les hypothèses.)
3. Combien de bits sont nécessaires après codage de Huffman ?

Exercice 23. (Examen 2007) On considère le code linéaire de correction d'erreur décrit par le tableau ci-dessous :

mots source	mots code
000	?00??
001	00101
0??	010??
011	??1?1
100	1001?
101	101?1
110	1100?
111	111??

1. En utilisant l'hypothèse de linéarité, remplacer les "?" par les éléments binaires manquant ("0" ou "1").
2. Expliciter la matrice génératrice du code.
3. Quel est l'image d'une séquence $[abc]$, où a, b, c sont des nombres binaires dans $\mathbb{Z}/2\mathbb{Z}$?
4. En déduire deux relations simples caractérisant les composantes des mots du code.
5. On note les vecteurs sous forme de colonnes. Expliquer pourquoi la matrice de contrôle permettant de générer le syndrome est de taille 2.
6. Déduire des trois questions précédentes une matrice de contrôle.
7. Combien ce code corrige-t-il d'erreurs ?

Exercice 24. (Examen 2006) Une source S sans mémoire délivre des 0 avec une probabilité de $p = 0,98$ et des 1 avec une probabilité de $1 - p = 0,02$. Le débit est de 300kbits/sec. La transmission se fait à travers un canal symétrique de probabilité d'erreur par élément binaire $0,05$ et fonctionnant avec un débit de 290kbits/sec.

1. Peut-on envisager d'utiliser ce canal pour transmettre le contenu de S avec une probabilité d'erreur aussi petite que souhaitée ?
2. Codage Source.
   On se propose de réduire le débit binaire de la source d'au moins 50% grâce à un code de Huffman. On suppose que les bits produit par la source S sont ensuite traités par le codeur Huffman qui définit ainsi une sourceS'.
   (a) Calculer l'entropie de la source S.
   (b) Quel doit être l'ordre minimum de l'extension de la source S permettant d'assurer une telle performance ?
   (c) Construire un arbre de Huffman associé à l'extension 3 de la source.
   (d) Quelle est la valeur du débit moyen de la nouvelle binaire obtenue à partir du codage précédent ?
   (e) Combien doit-on ajouter de bits de contrôle par bit émis par la nouvelle source S' si on veut utiliser le canal à son débit nominal[1] de 290kbits/sec.

[1] nominal=optimal

3. Codage canal.
   On souhaite maintenant coder la source binaire S' construite par algorithme de Huffman dans la partie précédente afin de réduire le taux d'erreur dues à la transmission à travers le canal. Supposons que l'on ajoute à chaque couple de bits émis par S' trois bits de contrôle.
   (a) Si on exige du code qu'il corrige une erreur par mot (de 5 bits), quelle doit être la valeur minimum de sa distance minimum ?En déduire le nombre minimal de 1 dans les mots du code.
   (b) Construire la matrice permettant de calculer un syndrome qui indique la position d'une erreur dans la 5 bits.
   (c) En déduire le code en indiquant pour chaque mot de S' le mot codé correspondant.
   (d) Construire la table de décodage, c'est-à-dire le tableau qui indique l'opération de correction en fonction du syndrome observé. Combien de configuration à deux erreurs permet-elle de corriger ? (Compter les syndromes qui ne peuvent être obtenus avec une seule erreur)
   (e) Calculer la probabilité d'erreur par mot code résultant de ce codage. Comparer cette valeur avec celle qui serai obtenue si on transmettait directement (i.e. sans codage les mots de l'extension d'ordre deux de la source S'.

Exercice 25. (Examen de rattrapage 2006) On souhaite compresser par un code de Huffman un texte composé de sept symboles dont les fréquences sont présentées dans le tableau suivant :

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ 0,49 & 0,26 & 0,12 & 0,04 & 0,04 & 0,03 & 0,02 \end{pmatrix}$$

On considère un codage direct des symboles du texte. Construire l'arbre de Huffman correspondant aux fréquences indiquées dans le tableau. Calculer le taux de compression que l'on peut espérer.

Exercice 26. (Examen de rattrapage 2006) On considère le code correcteur défini par $\phi(a1a2a3) = c1c2c3c4c5c6$ avec :

$$c1 = a1, c2 = a2, c3 = a3, c4 = a1 + a2, c5 = a2 + a3, c6 = a1 + a3.$$

1. Calculer la distance minimale entre deux symboles du code.
2. On considère le cas où une seule erreur survient. Montrer que ce code corrige cette erreur dans un cas sur 6.
3. Justifiez que ce code est un code globalement meilleur qu'un code de correction qui répète une fois chaque séquence de 3 symboles.

Exercice 27. (Examen de rattrapage 2007) On considère l'algorithme de compression d'Huffman binaire.

1. Pourquoi dans un code de compression sans perte optimal, les deux symboles de probabilités les plus faibles sont codées avec le même nombre de bits ?
2. On considère une source émettant cinq symboles $a_1, ..., a_5$ avec les probabilités suivantes :

Symbole	*Proba.*
a_1	$0,2$
a_2	$0,4$
a_3	$0,2$
a_4	$0,1$
a_5	$0,1$

   (a) Construire un arbre en suivant l'algorithme de Huffman et donner un codage correspondant.
   (b) Montrer, par un exemple, que l'on peut obtenir plusieurs arbres différents en appliquant l'algorithme.
   (c) La variance des tailles des symboles de codes obtenues avec deux arbres différents est-elle constante ? Justifier brièvement votre réponse.
   (d) A votre avis, pourquoi choisit-on préférentiellement le codage qui produit la plus petite variance ?

Exercice 28. (Examen de rattrapage 2007) Pour détecter et corriger les erreurs produites par un canal de transmission de signaux binaires, on décide d'envoyer trois fois de suite chaque bit du message à envoyer. On appelle *stratégie de triplement* cette méthode.

1. Donner les caractéristiques d'une telle stratégie :
   (a) Comment détecter une erreur en sortie du canal ?
   (b) Est-ce un codage par bloc ?
   (c) Quelle est la taille des blocs ?
   (d) Combien détecte-t-elle d'erreur par bloc ?
   (e) Combien corrige-t-elle d'erreur par bloc ?
   (f) Quelle est la distance du code correcteur associé à cette stratégie ?
2. On suppose que le canal produit une erreur avec une probabilité p. Si on utilise la stratégie de triplement, quelle est la probabilité qu'un bit du message initial (c'est-à-dire avant codage par la stratégie de triplement) soit erroné en sortie (c'est-à-dire après décodage) de canal ?
3. Quel est l'inconvénient de cette stratégie ?

Exercice 29. (Examen 2008) On considère un alphabet A comprenant s symboles. Pour tout mot w appartenant A^n, on définit :
- la boule de centre w et de rayon k : $B(w,k) = \{v \in A^n; d(w,v) \leq k\}$
- la sphère de centre w et de rayon k : $S(w,k) = \{v \in A^n; d(w,v) = k\}$

1. Exemple : $s = 2$, soit $w = 0110 \in \{0,1\}^4$, déterminer $S(w,1), S(w,2)$, et $B(w,2)$.
2. Déterminer le nombre d'éléments de $S(w,k)$.
3. Montrer que

$$\text{Card}\,(B(w,k)) = C_n^0(s-1)^0 + C_n^1(s-1)^1 + C_n^2(s-1)^2 + ... + C_n^k(s-1)^k = \sum_{i=0}^{k} C_n^i(s-1)^i.$$

4. Soit $C \in A^n$ un code correcteur des mots de A^p (bien sûr $p < n$) et de distance d. On note $t = E((d-1)/2)$: c'est le nombre d'erreurs que peut corriger C.
   (a) Quel est le nombre de mots de C ?
   (b) Montrer que si w_1 et w_2 sont deux mots distincts de C, alors $B(w_1,t) \cap B(w_2,t)$ est vide.
   (c) En déduire l'inégalité de Hamming :

$$\sum_{i=0}^{t} C_n^i(s-1)^i \leq s^{n-p}.$$

5. Applications : On appelle code parfait un code tel qu'on ait l'égalité

$$\sum_{i=0}^{t} C_n^i(s-1)^i = s^{n-p}.$$

   Dans ce cas, les boules $B(w,t), w \in C$, forment une partition de A^n. Dans les questions qui suivent, on considère seulement le cas $s = 2$.
   (a) Montrer que le code binaire par triplement ($n = 3, p = 1$) et de distance 3 est un code parfait. Même chose pour le code binaire de Golay de taille $n = 23, p = 12$ et de distance 7.
   (b) Soit C un code binaire de taille $n = 12, p = 5$. En utilisant l'inégalité de Hamming, donner une majoration de sa distance.

8.4 Chapitre 4 – Transformation de Fourier discrète

Exercice 30. (Phénomène du renversement binaire) On considère un échantillon de $N = 2^3$ valeurs $a_0, ..., a_7$.

1. Écrire les étapes successives de l'algorithme FFT. Quels sont les groupements successifs de coefficients a_i considérés au cours de l'algorithme ?
2. Comparer les écritures binaires des indices de la suite obtenue en concaténant les paquets du dernier appel récursif de l'algorithme. Que remarque-t-on ?
3. Généraliser.

Exercice 31. (Produit de polynômes de haut degrés) On considère deux manières de représenter un polynôme P de degré $N = 2^s$: la représentation par la suite de ces coefficients et la représentation "paire-valeur" $\{(x_1, y_1), ..., (x_m, y_m)\}$, avec $x_i \neq x_j$ pour $i \neq j$ et $P(x_j) = y_j$ pour tout j.

1. Dans quel cas les deux représentations sont équivalentes ? Comment passer d'une représentation à l'autre ?
2. Expliquer pourquoi la représentation paire-valeur permet un calcul rapide du produit de deux polynômes.
3. En déduire un algorithme de coût $O(N \log_2(N))$ permettant le calcul du produit de deux polynômes.

Exercice 32. (Coefficients de la TFD et approximation) Soit s un signal T-périodique et N un entier pair. Les coefficients de Fourier de s sont donnés par la formule :

$$\hat{s}_n = \frac{1}{T} \int_0^T s(t) \mathrm{e}^{-i2\pi n \frac{t}{T}} dt.$$

Écrire l'approximation $\hat{s}'_n$ obtenue par la formule des trapèzes.

Calculer les coefficients du polynôme trigonométrique

$$p(t) = \sum_{n=-N/2}^{N/2-1} c_n^N \mathrm{e}^{i2\pi n \frac{t}{T}}$$

qui interpole s aux points k/NT. On posera par commodité :

$$\begin{cases} Y_n = c_n^N & si\ 0 \leq n \leq N/2 - 1 \\ Y_n = c_{n-N}^N & si\ N/2 \leq n \leq N - 1 \end{cases}$$

Que remarque-t-on ?

Exercice 33. (FFT en base 4) Trouver un algorithme équivalent à celui présenter en cours dans le cas binaire pour le cas $N = 4^s$.

8.5 Chapitre 5 – Filtres numériques

Exercice 34. Un système linéaire numérique donne la réponse :

$$y_n = \{122\}$$

lorsqu'il subit l'entrée :

$$x_n = \{103\}.$$

1. Calculer la fonction de transfert du système.
2. Le système est-il stable ?
3. Calculer la réponse impulsionnelle.

Exercice 35. Déterminer les transformées en z des fonctions de transfert suivantes, ainsi que leurs domaines de définition :

1. $h(n) = 1$ pour $n \geq 0$ et 0 ailleurs,
2. $h(n) = 1$ pour $0 \leq n \leq N$ et 0 ailleurs,
3. $h(n) = a^n$ pour $n \geq 0$ et 0 ailleurs,
4. $h(n) = \cos(\omega n)$ pour $n \geq 0$ et 0 ailleurs,
5. $h(n) = a^n \sin(\omega n)$ pour $n \geq 0$ et 0 ailleurs,
6. $h(n) = na^n$pour $n \geq 0$ et 0 ailleurs.

Exercice 36. Donner la réponse impulsionnelle d'un filtre dont la réponse fréquentielle est donnée par la formule :

$$H(\mathrm{e}^{i\omega}) = \exp\big(\cos(\omega)\big)\mathrm{e}^{i\sin(\omega)}.$$

Exercice 37. Déterminer la réponse en fréquence des filtres (stables) suivants :

1. $y_n = \frac{x_n + x_{n-1}}{2}$,
2. $y_n = \frac{1}{2}y_{n-1} + x_n$.

Exercice 38. On considère un filtre de Bernouilli dont la réponse impulsionnelle est définie par $h_0 = p$, $h_1 = q$ et $h_n = 0$ ailleurs (p et q sont fixés tels que $p + q = 1$).

1. Calculer $H(z)$,
2. Calculer la réponse impulsionnelle d'un filtre binômial d'ordre n obtenu en composant n fois le même filtre de Bernouilli ;
3. Les filtres binômiaux sont-ils RIF, causaux, stables ?
4. Calculer leur réponse fréquentielle. Tracer grossièrement l'allure de $|H(\omega)|$.
5. Vérifier que ces filtres sont des filtres passe-bas.

Exercice 39. On considère la fonction de transfert :

$$H(z) = \frac{1 + 3z^{-1}}{2 - z^{-1}}.$$

1. Rappeler l'équation reliant une entrée x_n à la sortie y_n ?
2. Caractériser le filtre : RII, RIF ? Comportement en fréquence ?

Exercice 40. On associe au signal numérique x le signal numérique y de la manière suivante :

$$y_n = y_{n-1} + x_n.$$

1. Montrer que ce filtre est linéaire et invariant par translation.
2. Calculer la réponse impulsionnelle h associée à ce filtre.
3. Ce filtre est-il stable ?
4. Calculer la fonction de transfert.
5. Calculer la réponse en fréquence.

Exercice 41. On considère le filtre numérique caractérisé par :

$$H(z) = K(1 - z^{-10}).$$

1. Donner l'algorithme de filtrage et la réponse impulsionnelle de ce système.
2. Ce filtre est-il stable ? Pourquoi ?
3. Ce filtre est-il à phase linéaire ? Calculer la phase.
4. Étude de la réponse en fréquence du système.
   (a) Calculer la réponse en fréquence $|H(\mathrm{e}^{i2\pi f})|$ pour $0 \leq f \leq 1/2$.
   (b) Déterminer les fréquences f_{max} et f_{min} pour lesquelles on obtient H_{min} et H_{max}, les valeurs minimales et maximales du module de la fonction de transfert.

(c) Déterminer la valeur de K qui normalise H_{max} à 1.

(d) Dessiner l'allure de $e^{i2\pi f}$.

(e) Déterminer les pôles et les zéros et justifier les résultats précédents.

Exercice 42. Donner une méthode de calcul d'un filtre passe-bande correspondant à deux fréquences de coupures a et b.

Exercice 43. On considère le filtre correspondant à la fonction de transfert $H(z) = 1 + z^{-1} + z^{-2} + ...$

1. Montrer que ce filtre non-récursif peut être remplacé par un filtre récursif très simple.
2. Déterminer le comportement fréquentiel de ce filtre et tracer les graphes $f \mapsto |H(e^{i2\pi f})|$ et $f \mapsto arg(H(e^{i2\pi f}))$.

Exercice 44. (Examen de rattrapage 2006) Déterminer et représenter les réponses en amplitude et en phase des filtres suivants :

1. $y(n) = \frac{1}{2}\big(x(n) + x(n-1)\big)$,
2. $y(n) = \frac{1}{4}\big(x(n) + x(n-1) + x(n-2) + x(n-3)\big)$,
3. $y(n) = \frac{1}{4}\big(x(n) - 2x(n-1) + x(n-2)\big)$.

Exercice 45. (Examen de rattrapage 2006) On considère un filtre linéaire de réponse impulsionelle $h = (h_n)_{n\in\mathbb{Z}}$. On note $x = (x_n)_{n\in\mathbb{Z}}$ le signal à l'entrée du filtre et $y = (y_n)_{n\in\mathbb{Z}}$ le signal en sortie.

1. Rappeler la définition de la réponse impulsionnelle.
2. Montrer que les suites x, y et h sont liées par la formule :
$$y = h \star x,$$
où $\star$ désigne le produit de convolution discret.
3. Rappeler la définition de la transformée en z d'un signal numérique $u = (u_n)_{n\in\mathbb{Z}}$.
4. Montrer que les transformées en z des signaux x, y et de la réponse impulsionnelle h vérifient :
$$H(z) = \frac{Y(z)}{X(z)}.$$
5. On considère le filtre défini par la relation :
$$y_n = \frac{1}{2}y_{n-1} + 2x_n.$$
(a) Calculer la réponse impulsionnelle h de ce filtre et $H(z)$, sa transformée en z.

(b) Ce filtre est-il stable ?

(c) Quelle est sa réponse fréquentielle ? Le filtre est-il plutôt passe-haut ou plutôt passe-bas ?

Exercice 46. (Examen 2008)
Dans cet exercice, on étudie un filtre donné par un schéma-bloc.

1. On considère tout d'abord une sous-partie du filtre. Cette sous-partie est le filtre représenté par le schéma-bloc représenté Fig. 8.1. Le bloc " $\times g$" représente la multiplication par un réel g donné.
Donner, en fonction de g, l'expression de la fonction de transfert $H'(z)$ de cette sous-partie du filtre.
2. On considère maitenant le filtre complet, repésenté par le schéma-bloc représenté Fig. 8.2.
Le bloc " $\times b$" représente la multiplication par un réel b donné. Donner, en fonction de a, g et b, l'expression de la fonction de transfert $H(z)$ du filtre complet.
3. Sous quelle condition le filtre complet est-il stable ?
4. On suppose que les coefficients du filtre vérifient les relations :
$$a = 1 - g^2,\ b = -g.$$
Montrer que le module de la réponse fréquentielle est constant.
5. Pourquoi, à votre avis, appelle-t-on ce filtre "réverbérateur" ?

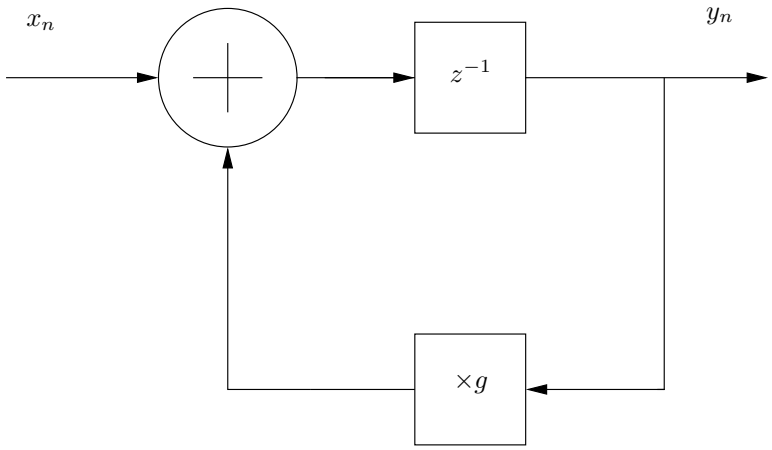

FIG. 8.1 – Sous-partie du filtre

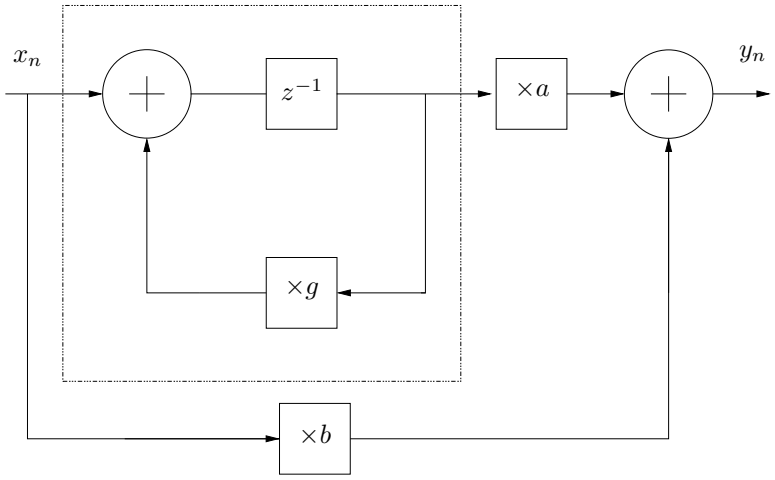

FIG. 8.2 – Filtre complet

8.6 Chapitre 6 – Conception de filtres numériques

Exercice 47. (Examen 2008) On veut réaliser un filtre passe-bas numérique à réponse impulsionnelle finie à $N = 17$ coefficients. La fréquence de coupure est : $fc = fe/4$ avec $fe = 1$ fréquence d'échantillonnage. On suppose que le théorème de Shannon est respecté.

1. Donner l'allure de la réponse en fréquence $H(\omega)$ sur $[-\pi, \pi]$ du filtre idéal non causal à phase nulle correspondant au cahier des charges.
2. Déterminer $(\tilde{h}_k)_{k\in\mathbb{Z}}$ la réponse impulsionnelle du filtre idéal à phase nulle.
3. Effectuer la troncature puis rendre causal cette réponse impulsionnelle et donner les coefficients du filtre finalement obtenu.
4. Donner l'expression du module de la réponse fréquentielle du filtre obtenu en fonction des coefficients. Esquisser l'allure de la réponse fréquentielle.
5. Quels sont les effets de la troncature de $(h_k)_{k\in\mathbb{Z}}$ sur la réponse en fréquence du filtre synthétisé ? Comment pourrait-on améliorer les résultats ?

Exercice 48. (Synthèse de filtres RII par approximation de Padé – Rattrapage 2006) On souhaite concevoir un filtre de réponse impulsionnelle $h_d(n), n \geq 0$. On choisit a priori un filtre dont la fonction de transfert est de la forme :

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = \sum_{k=0}^{\infty} h(k) z^{-k}.$$

La fonction $H(z)$ a donc $L = M + N + 1$ paramètres (les coefficients $\{a_k\}$ et $\{b_k\}$) à déterminer.
Supposons que l'on mette en entrée de notre filtre une impulsion de Dirac $x(n) = \delta(n)$ (c'est-à-dire, un "1" suivi de zéros).

1. Montrer que la réponse est alors de la forme :

$$\begin{aligned} h(n) &= -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N) \\ &\quad + b_0 \delta(n) + b_1 \delta(n-1) + ... + b_M \delta(n-M). \end{aligned} \tag{8.1}$$

2. Montrer que (8.1) peut également s'écrire :

$$h(n) = -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N) + b_n, \ 0 \leq n \leq M. \tag{8.2}$$

3. Montrer que pour $n > M$, l'équation (8.2) peut encore se simplifier en :

$$h(n) = -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N), \ n > M. \tag{8.3}$$

4. Expliquer comment les équations (8.2) et (8.3) peuvent être utilisées pour déterminer $\{a_k\}$ et $\{b_k\}$ si l'on impose $h(n) = h_d(n)$ pour $0 \leq n \leq N + M$.
5. Quelles sont les limitations de la méthode ? Comment y remédier ?

Exercice 49. On considère le filtre défini par :

$$H(z) = \frac{1 + 3z^{-1}}{2 - z^{-1}}$$

.

1. Calculer la réponse impulsionelle.
2. On veut implanter ce filtre par une structure RIF équivalente. Pour cela on tronque la réponse impulsionnelle. Les valeurs de $h(n)$ sont codées en binaire sur huit bits, selon le format bit $n \leftrightarrow coef.\ de\ 2^{-n}$.
   (a) A quel rang est-il naturel de tronquer ?
   (b) Quelle est la réponse impulsionnelle ?

Exercice 50. (Examen 2007)

1. Analyse d'un filtre numérique
   On considère un système numérique caractérisé par :

$$H(z) = K.[1 - z^{-10}]; K > 0$$

   (a) Donner l'algorithme de filtrage et la réponse impulsionnelle de ce système.
   (b) Ce filtre est-il stable ? justifier votre réponse.
   (c) Est-il à phase linéaire ? justifier votre réponse.
   (d) Réponse en fréquence du système :
      i. Calculer $|H(\mathrm{e}^{i\omega}|$ et $j(\omega) := arg\left(H(\mathrm{e}^{i\omega})\right)$.
      ii. Déterminer les fréquences f_{max} et f_{min} pour lesquelles on obtient les valeurs respectivement maximales (H_{max}) et minimales (H_{min}) de $|H(\mathrm{e}^{i\omega})|$.
      iii. Déterminer la valeur de K qui normalise H_{max} à l'unité.

iv. Dessiner l'allure de $|H(e^{i\omega})|$ pour cette valeur de K.

2. Synthèse d'un filtre RIF passe-bas

On désire réaliser un filtre numérique passe-bas dont la fréquence de coupure est $f_e/3$, où f_e est la fréquence d'échantillonnage.

(a) Donner le graphe de la réponse en fréquence $G(f)$ du filtre idéal à phase nulle répondant a ce cahier des charges.

(b) Déterminer la réponse impulsionnelle g_k du filtre idéal non causal à phase nulle par développement en séries de Fourier de $G(f)$.

(c) On décide de fixer la longueur du filtre à N termes. Montrer que ce choix permet de rendre le filtre RIF causal et donner l'algorithme de filtrage.

8.7 Chapitre 7 – Codage en sous-bandes

Exercice 51. On considère un filtre de fonction de transfert $H(z) = \sum_{n\in\mathbb{Z}} h_n z^{-n}$.

1. Montrer que le filtre peut se mettre sous la forme :

$$H(z) = H_0(z^2) + z^{-1}H_1(z^2),$$

et donner les réponses impulsionnelles des filtres H_0 et H_1.

2. Calculer les fonctions H_0 et H_1 dans le cas où :

$$H(z) = \frac{1}{1 - az^{-1}}.$$

Exercice 52. On considère banc QMF de filtres RIF. On cherche sous quelle condition on a reconstruction parfaite. De manière à garantir la condition de non-repliement, on impose :

$$F_0(z) = H_1(-z),\ F_1(z) = -H_0(-z),$$

et on cherche à satisfaire :

$$T(z) := H_0(z)F_0(z) + H_1(z)F_1(z) = 2z^{-n_0}.$$

1. En décomposant H_0 sous la forme :

$$H_0(z) = H_{00}(z^2) + z^{-1}H_{01}(z^2),$$

montrer que $T(z)$ s'écrit sous la forme :

$$T(z) = 2z^{-1}H_{00}(z^2)H_{01}(z^2).$$

2. En déduire qu'on aura reconstruction parfaite si :

$$H_{00}(z) = h_{00}z^{-k},\ H_{01}(z) = h_{01}z^{-k'}.$$

3. En déduire que les seuls filtres RIF générant des bancs QMF assurant une reconstruction parfaite sont de la forme :

$$H_0(z) = \alpha z^{-2k} + \beta z^{-(2k'+1)}.$$

Chapitre 9

Travaux pratiques

9.1 Révisions Matlab

Exercice 1.
Tracer des graphes pour illustrer le phénomène de Gibbs de l'exercice 1. Regarder ce qui se passe si l'on considère les sommes de Féjer au lieu des sommes de Dirichelet.
Exercice 2.
Étant donné $n > 0$, écrire en utilisant le moins de lignes possible un script Matlab qui déclare la matrice de Cauchy de coefficients
$a_{i,j} = \frac{1}{i+j}$.
Exercice 3.
A l'aide de *rand*() tracer une Gaussienne.

9.2 TP1 : Quantification

9.2.1 Quelques commandes Matlab

On commence par quelques manipulations de fichiers à l'aide de Matlab.

Lecture d'une image

On prendra comme image de référence celle téléchargeable ici :
http ://docs.ufrmd.dauphine.fr/optinum/data/patchwork.tif

1. Comprendre comment charger et afficher une image en Matlab. On utilisera les commandes matlab suivantes :
   `readim, imagesc, colormap`
   Pour les questions suivantes, on utilisera les commandes `imfinfo` et `size`.
2. Quel est le type du fichier obtenu ?
3. Comment interpréter son contenu ?
4. Qu'en déduit-on sur le codage de l'image ?
5. Quelle est la taille du fichier attendue ? La comparer à la taille du fichier téléchargé.

Conversion binaire/décimal et taille des données

Il va nous être utile dans la suite (et plus généralement dans les projets) de traiter des données binaires.

1. Convertir l'image en un fichier binaire.
   On utilisera les commandes matlab suivantes :
   `dec2bin`

2. Comparer les tailles des deux fichiers.
   On utilisera la commandes matlab suivante :
   `whos`
   Remarque : vous constaterez que deux bits sont en fait utilisés pour coder un bit.
3. A partir du fichier binaire, reformer le fichier image initial.
   On utilisera les commandes matlab suivantes :
   `bin2dec, reshape`

9.2.2 Compression par quantification

Dans cette partie, on quantifie de manière naïve le fichier.

1. Effectuer une compression par quantification naïve en codant l'image sur 3 bits. On utilisera le fichier binaire associé à l'image.
2. Regarder et interpréter la taille annoncée du fichier.
3. Visualiser l'image quantifiée.

9.3 TP2 : Conception de filtres

9.3.1 Filtrage RIF

Un filtre non récursif est défini par la relation de récurrence :

$$y_n = x_n + a x_{n-p}.$$

On choisit ici $a = 1$ et $p = 5$.

1. Quelle est la réponse impulsionnelle du filtre ?
2. Afficher sous matlab le module de la réponse en fréquence du filtre.
3. Afficher sous matlab la phase de la réponse en fréquence du filtre.

9.3.2 Synthèse d'un filtre RIF passe-bas

On veut réaliser un filtre RIF passe-bas de fréquence de coupure $Fe/5$. On va utiliser la méthode de la fenêtre. Pour cela, on commence par calculer la réponse impulsionnelle correspondant au filtre passe-bas idéal, par transformée de Fourier à temps discret inverse.

1. Quelle est la réponse impulsionnelle du filtre passe-bas idéal de fréquence de coupure $Fe/5$ et de phase nulle ?
2. Afficher sous matlab la réponse impulsionnelle causale de longueur 81 échantillons obtenue par troncature et décalage de la réponse impulsionnelle idéale.
3. Afficher le module (sans unité puis en en dB) de la réponse en fréquence correspondante, en utilisant une TFD sur $N = 1000$ points fréquentiels.

Le filtre obtenu est-il passe-bas ? Vérifier que la fréquence de coupure est bien la bonne. Quel est le défaut de ce filtre passe-bas ?

1. Refaire l'opération mais au lieu de tronquer (ce qui correspond implicitement à utiliser une fenêtre rectangulaire), utiliser une fenêtre de Blackman[1]. Afficher le module (en dB) de la réponse en fréquence.
2. Qu'a-t-on amélioré par rapport au cas précédent ?
3. Qu'a-t-on détérioré ?

[1] Celle-ci est définie par $w = .42 - .5 * \cos(\frac{2*\pi*n}{M-1}) + .08 * \cos(\frac{4*\pi*n}{M-1})$;

9.3.3 Filtrage RII

Un filtre récursif est défini par la relation de récurrence :

$$y_n - 0.95y_{n-5} = x_n.$$

Quelle est la fonction de transfert $H(z)$ du filtre ? Les filtres de fonction de transfert rationnelle du type

$$H(z) = B(z).A(z)$$

peuvent être représentés sous matlab par deux vecteurs : le vecteur correspondant au polynôme en z^{-1}, $A(z)$ et le vecteur correspondant au polynôme en z^{-1}, $B(z)$. Matlab permet de filtrer n'importe quel signal par un filtre de ce type grâce à la fonction 'filter' : Si x est un signal, $y = filter(B, A, x)$ calcule la sortie du filtre défini par la récurrence ci-dessus.

1. Sous matlab, en utilisant la fonction filter, calculer et afficher la réponse impulsionnelle du filtre correspondant à la récurrence ci-dessus.
2. Calculer et afficher la réponse en fréquence correspondante.
3. Comparer avec :

```
Bf= fft(B,512);
Af= fft(A,512);
Hf= Bf./Af;
plot(abs(Hf(1:256))
```

```
%%%%%%  Filtrage RIF  %%%%%%%%
a=.9;p=5;
N=20;
x=zeros(1,N);
y=zeros(1,N);
x(1)=1;
for  k=1:N
if k>p
y(k)=x(k)+a*x(k-p);
else
y(k)=x(k);
end
end

I=0:.01:.5;
A=exp(i*2*pi*I);
rep_mod=abs(1+a*A.^(-p));
rep_phase=angle(1+a*A.^(-p));

%%%%%%%  Filtrage RIF. Synthèse d'un filtre passe-bas  %%%%%%%%
%Réponse impulsionnelle causale
fc=1/5;wc=2*pi*fc;
M=81;
h=zeros(1,M);

h=1/pi*sin(wc*([0:M-1]-(M-1)/2))./([0:M-1]-(M-1)/2);
h((M-1)/2+1)=wc/pi;

%Réponse en fréquence
N=1000;
```

```
I=(linspace(0,.5,N))';
a=exp(i*2*pi*I);
H=sum(diag(h)*((ones(M,N)*diag(a)).^(diag([0:M-1])*ones(M,N))));
%plot(linspace(0,.5,N),abs(H));

%Fenêtre de Blackman
J=0:M-1;
Blackman=.42-.5*cos(2/(M-1)*pi*J)+.08*cos(4/(M-1)*pi*J);
h_new=h.*Blackman;
H_new=sum(diag(h_new)*((ones(M,N)*diag(a)).^(diag([0:M-1])*ones(M,N))));
%plot(linspace(0,.5,N),10*log(abs(H_new)),linspace(0,.5,N),10*log(abs(H)));

%%%%%%%  Filtrage RII  %%%%%%%%
%Réponse impulsionnelle
B=[1 0 0 0 0];
A=[1 0 0 0 -.95];
l=100;
x=[1, zeros(1,l-1)];
y=filter(B,A,x);
%Réponse en fréquence
rf=1./(1-.95*a.^(-5));
%plot(linspace(0,.5,N),abs(rf));
```

Bibliographie

[1] Pierre Brémaud, *Mathematical principles of signal processing*, Springer-Verlag, 2002.

[2] Khalid Sayood, *Introduction to data compression,* 3^{rd} *ed.*, Morgan Kauffmann-Elsevier, 2006.

[3] John G. Proakis, Dimitri Manolakis, *Digital signal processing,* 4^{th} *ed.*, Pearson-Prentice Hall, 2007.

[4] Thomas M. Cover, Joy A. Thomas, *Element of information theory,* 2^{nd} *ed.*, Wiley, 2006.

[5] Etienne Tisserand, Jean-François Pautex, Patrick Schweitzer *Analyse et traitement des signaux*, Dunod, 2004.

Index