IDIAP COMMUNICATION

IDIAP
Martigny - Valais - Suisse

ANNULATION D'ÉCHO SUR UNE LIGNE TÉLÉPHONIQUE

Florian Salamin [a] François Corthay [a]
Olivier Bornet [b] Jean-Luc Cochard [b]

IDIAP–COM 96-06

DÉCEMBRE 1996

Institut Dalle Molle
d'Intelligence Artificielle
Perceptive • CP 592 •
Martigny • Valais • Suisse

téléphone +41−27−721 77 11
télécopieur +41−27−721 77 12
adr.él. secretariat@idiap.ch
internet http://www.idiap.ch

---

[a] Ecole d'Ingénieur du Valais rue du Rawyl 47 1950 Sion

[b] Institut Dalle Molle d'Intelligence Artificielle Perceptive Case Postale 592 1920 Martigny

École d'Ingénieurs du Valais (EIV) Ingenieurschule Wallis (ISW)

Département électrotechnique

Travail de diplôme 1996
Florian Salamin

Annulation d'écho sur une ligne téléphonique

Professeur: François Corthay
Mandant: IDIAP, Martigny
Expert: Jean-Luc Cochard

Sion, le 6 décembre 1996

SALAMIN FLORIAN DIPLÔME 1996

ANNULATION D'ÉCHO SUR UNE LIGNE TÉLÉPHONIQUE

OBJECTIFS

Ce travail a pour cadre les serveurs vocaux interactifs avec reconnaissance de la parole. La création d'outils logiciels doit permettre la mise en évidence les phénomènes d'écho sur une ligne téléphonique. L'analyse et le développement d'une méthode d'annulation de cet écho doit faciliter la mise en oeuvre de la reconnaissance vocale.

RÉSULTATS

Les outils nécessaires à l'étude de l'écho sont crées et sont opérationnels. L'écriture de l'application d'annulation de l'écho est bien avancée. Elle permet la vérification du concept d'annulation sur des signaux idéaux. L'expérimentation avec des signaux réels n'est pas encore terminée.

MOTS CLÉS

Transformée de Fourier, C et C++, autocorrélation et corrélation croisée, traitement de signal, RNIS, loi A.

ZIELE

Diese Arbeit ist im Bereich interaktiver Sprachserver mit Erkennung der Sprache angesiedelt. Die Phänomene des Echos auf einer Telefonleitung müssen mit Software-Hilfsmitteln aufgezeigt werden. Die Analyse und Entwicklung einer Methode zur Aufhebung des Echos muss die Inbetriebnahme einer Spracherkennung vereinfachen.

RESULTATE

Die notwendigen Hilfsmittel zum Studium des Echos wurden erzeugt und funktionieren. Die Programmierung der Anwendung zur Aufhebung des Echos ist gut fortgeschritten. Die Tests mit idealen Signalen bestätigen die Richtigkeit des Konzeptes. Die Experimentation mit reellen Signalen wurde noch nicht abgeschlossen.

SCHLÜSSELWÖRTER

Fouriertransformation, C und C++, Autokorrelation und Kreuzkorrelation, Signalverarbeitung, ISDN, A-law.

Note au lecteur

Plusieurs conventions ont été utilisées pour la rédaction de ce document. C'est notamment le cas pour:

- les références bibliographiques qui sont signalées dans le texte par un chiffre en italique entre crochets *[x]*. La référence en question renvoie à la bibliographie à la fin du document.
- les annexes au document qui sont fournis sur support magnétique, se trouvent sur une disquette au format DOS. Dans le texte, le nom indiqué en italique correspond au nom du fichier.

Table des matières

Introduction 1

- Les serveurs vocaux interactifs 1
- Développement des SVIs en Suisse 1
- Les SVIs au niveau mondial 2

Cahier des charges 3

Planning 4

Contexte 5

- L'institut IDIAP à Martigny 5
- Station de travail UNIX 5
- Programmation en C et C++ 5
- Le RNIS 6

Origine de l'écho 7

Un mot sur le projet de semestre 9

- Implémentation full-duplex 9
- SunXTL en bref 10
- Adaptations du logiciel 11
- Analyse des solutions 12

Concepts 13

- Hypothèses de travail 13
- Atténuation et décalage dans le temps 13
- La corrélation 14

Présence de l'écho 21

Signaux utilisés 22

- Extrait de voix 23
- La fonction sinus cardinal 24

Évaluation des méthodes de calcul 25

DFT et FFT 26
Corrélation dans le temps 28
Corrélation dans la fréquence 29
Résultats 31

Écriture du programme 32

Découpage du logiciel 32
La classe cfft 34
Les modules 35
Programmes auxiliaires 40
La “makefile” 41

Tests 42

Code source 42
Correction de signaux idéaux 43
Correction de signaux réels 46

Résultats et propositions 50

Conclusion 52

Remerciements 53

Bibliographie 54

Ouvrages 54
Ressources Internet 56

1. Introduction

1.1 Les serveurs vocaux interactifs

De nos jours, l'accès rapide et efficace à l'information est non seulement un facteur de réussite pour les entreprises, mais aussi pour tout un chacun. De plus, la souplesse de consultation de l'information est primordiale.

Les SVIs présentent plusieurs avantages vis-à-vis de la technique traditionnelle de diffusion des informations. L'appelant peut dialoguer directement avec la machine et ainsi la piloter par choix successifs jusqu'à l'obtention de l'information désirée. Ces dernières sont en outre disponibles 24 heures sur 24 et 7 jours sur 7. Le coût de l'information est considérablement réduit du fait que seuls un ordinateur et un media de communication sont nécessaires. On peut ainsi réaliser des services à valeur ajoutée élevée.

La mise en oeuvre d'un SVI ne demande que peu de matériel et est relativement simple (figure 1). Un serveur vocal peut se présenter sous la forme d'un ordinateur relié au réseau téléphonique par une ligne numérique de type RNIS (Réseau Numérique à Intégration de Services). N'importe quel abonné du réseau peut appeler le serveur depuis un téléphone analogique ou numérique.

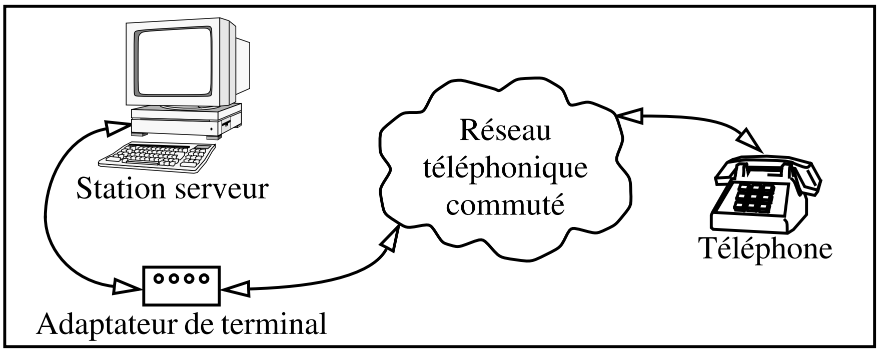


FIGURE 1. Principe de fonctionnement

La souplesse d'utilisation des SVIs est aussi un énorme avantage au niveau de la mise sur pied d'un service vocal d'information. L'implantation de serveurs vocaux se réalise en douceur car cet outil ne demande quasiment aucun apprentissage de la part de l'utilisateur. Il est aussi simple d'appeler un serveur vocal que de converser au téléphone avec un ami. Le SVI se greffe de manière “naturelle”.

1.2 Développement des SVIs en Suisse

L'avenir et le développement des serveurs vocaux interactifs en Suisse se présente de manière favorable. Des entreprises importantes acquièrent (PTT, UBS, SBS, Crédit Suisse) ou s'intéressent de près (Veillon) à cette technique. Il est intéressant de citer le service vocal d'horaire des trains mis sur pied par les CFF. Ce service est dès à présent disponible au numéro 157 02 22 *[14]*. Ce n'est guère qu'un marché naissant, mais le fait que les leaders de l'économie suisse y soient présents est un signe encourageant.

Les SVIs apportent à ces entreprises des avantages financiers indéniables ainsi qu'une augmentation de la productivité due à l'amélioration de la communication.

1.3 Les SVIs au niveau mondial

Le marché mondial est en pleine expansion. Aux États-Unis et en France, le taux de croissance du marché lié aux serveurs vocaux interactifs est de l'ordre de 20 à 30% par an. Tant en France qu'aux États-Unis, on acquiert des SVIs dans de nombreux secteurs d'activités économiques. Des banques, des sociétés de télécommunication, des sociétés de vente par correspondance, des offices du tourisme, des assurances, des services gouvernementaux, des instituts de météorologie et bien d'autres encore se lancent dans l'achat de serveurs vocaux interactifs.

En France, le marché du vocal interactif a généré un chiffre d'affaire de près de 400 millions de FS en 1992 et en Grande-Bretagne 500 millions de FS. Ce marché est considéré comme étant propice à une expansion importante à moyen terme.

2. Cahier des charges

L'IDIAP (Institut Dalle Molle d'Intelligence Artificielle Perceptive) à Martigny travaille sur la réalisation de serveurs vocaux interactifs. La société Sun Microsystems commercialise un environnement de développement de SVI appelé SunXTL.
Le principe de ce serveur vocal est une sorte d'arbre de décision qui se base sur un ensemble de mots-clefs. À chaque niveau, le nombre et la nature de ces mots-clefs changent pour guider l'utilisateur vers l'information désirée. La reconnaissance vocale se fait sur cet ensemble des mots-clefs.

Afin de rendre l'utilisation de ces serveurs plus conviviale, le locuteur et le serveur doivent pouvoir parler simultanément. Avant d'effectuer la reconnaissance de la parole, il faut mettre en oeuvre un mécanisme d'annulation de l'écho pour séparer correctement les signaux correspondants aux messages émis par le locuteur et le serveur.

Un premier travail, effectué lors du projet de semestre, m'a permis d'étendre les possibilités de la librairie *datapump* fournie avec l'environnement de développement SunXTL, en lui ajoutant la capacité de full-duplex. Cela signifie que cette librairie est maintenant capable de recevoir et de diffuser simultanément de l'information (voir “Un mot sur le projet de semestre” en page 9). La capacité full-duplex doit me permettre d'analyser les signaux émis et reçus pour quantifier l'écho.

Le but de ce projet est de mettre en évidence les phénomènes d'écho susceptibles d'apparaître lors d'une liaison entre le serveur vocal et un abonné relié au réseau téléphonique par une ligne analogique. Les résultats de ces mesures devront confirmer ou infirmer la pertinence de la mise en oeuvre de l'annulation d'écho au niveau du signal sonore numérisé provenant du locuteur.

Il faut aussi créer un programme qui doit permettre le calibrage de l'annulation d'écho. Ce programme doit, dans un intervalle de moins d'une seconde, extraire les paramètres qui serviront à la correction du signal reçu.

D'autre part, une étude des algorithmes de compression utilisés sur les lignes numériques sera indispensable à la mise en place correcte du traitement du flot de données provenant du réseau téléphonique. Dans le cadre de ce projet, c'est surtout la loi de compression européenne, connue sous le nom de loi A, qui est utilisée.

Le projet doit être conduit sur une station de travail Sun. Il me faut donc aussi acquérir un minimum de connaissance du système d'exploitation UNIX pour pouvoir mener à bien ce projet. Cela englobe aussi bien l'administration du système que l'écriture de scripts ou de programmes.

3. Planning

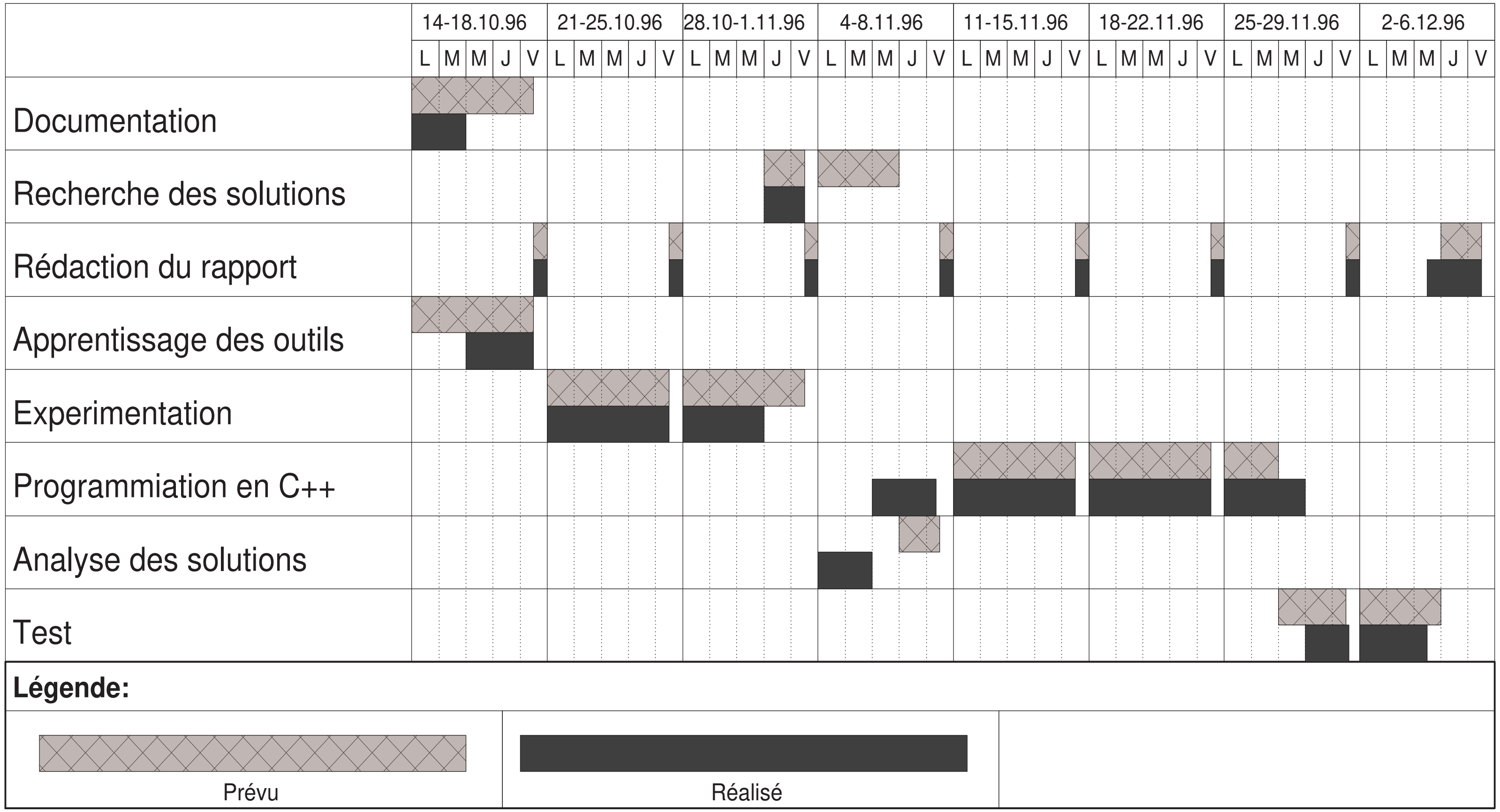


FIGURE 2. Planning

4. Contexte

4.1 L'institut IDIAP à Martigny

L'Institut Dalle Molle d'Intelligence Artificielle Perceptive (IDIAP) à Martigny dirige ses efforts dans les domaines des réseaux de neurones artificielles, de la vision artificielle et du traitement automatique de la parole. Dans ce dernier axe de recherche, l'IDIAP travaille aussi sur le développement de serveurs vocaux interactifs. L'un des objectifs est de mettre au point un serveur vocal que l'on peut piloter par la voix humaine. Il serait ainsi possible de s'affranchir de l'utilisation du clavier de téléphone pour commander le serveur d'informations. Mon travail de diplôme s'inscrit dans ce projet d'envergure mené par l'IDIAP. Il marque aussi la première collaboration entre l'École d'Ingénieurs du Valais et l'IDIAP.

4.2 Station de travail UNIX

Le domaine des serveurs vocaux, bien qu'en forte croissance, est encore marginal. De ce fait, bien peu de constructeurs d'informatique et de fournisseurs de logiciels disposent d'outils de développement adéquats. Sun Microsystems est l'une des sociétés qui proposent à la fois le hardware et le software nécessaires. Pour la mise en oeuvre de ce projet, il était indispensable de s'équiper en hardware. Sur les conseils avertis de M. Olivier Bornet, ingénieur de recherche à l'IDIAP, l'École a acheté une station de travail Sun de type Ultra1 équipée de 64 méga-bytes de RAM (Random Access Memory), de 5 giga-bytes de disque dur, et d'un écran de 17 pouces. De plus, l'École à fait l'acquisition d'un carte d'interface RNIS, du système d'exploitation Solaris 2.5, et du compilateur C++ SPARCWorks 4.1 de Sun. L'environnement de développement SunXTL est mis à disposition de l'École par l'IDIAP pour la durée du projet.

L'une de mes tâches importantes dans ce projet est d'acquérir la base de connaissances nécessaires avec le système d'exploitation UNIX pour pouvoir utiliser correctement la station de travail. J'ai aussi l'occasion de me familiariser quelque peu avec le système d'exploitation Solaris 2.5 et l'environnement graphique X Window.

4.3 Programmation en C et C++

Le langage de programmation utilisé pour ce projet est principalement le C++. Ce choix à été dicté par le fait que le logiciel de développement d'applications vocales proposé par Sun, ainsi que les exemples d'utilisation, ont été écrits en C++. Mes connaissances en programmation orientée objet doivent me permettre de saisir relativement rapidement le fonctionnement des méthodes de base de l'environnement SunXTL. Je dois, de plus, m'adapter au style de programmation C++ propre au monde

UNIX. Celui-ci diffère surtout quant à la surveillance du bon déroulement des différentes procédures.
Le langage C est quant à lui utilisé pour l'écriture de petits programmes utilitaires ou pour l'implémentation de modules et de fonctions.

4.4 Le RNIS

Le Réseau Numérique à Intégration de Services permet le transfert de données sur des lignes téléphoniques sous forme digitale. Un accès de base fournit à l'abonné 2x64kbit/s (deux canaux B) plus 1x16kbit/s (un canal D) en full-duplex. L'utilisation d'une ligne numérique est judicieuse. En effet, les données sur la ligne sont déjà transmises sous forme numérique. Elles sont donc directement lisibles par l'ordinateur. On évite ainsi, du côté du serveur, la numérisation du signal vocal.

L'interface RNIS installée dans la station de travail permet de relier l'ordinateur directement à la ligne numérique. Il devient ainsi possible de générer directement des données numériques sur la ligne. L'acquisition du flot d'entrée se fait de la même manière et il est ainsi possible de le mémoriser dans un fichier sur le disque dur de la station. L'environnement de développement fournit des outils pour piloter la carte au travers des *drivers* de celle-ci. Il est donc possible de prendre en charge un appel, de le mettre en attente, de le transférer, ou de raccrocher de manière logicielle.

5. Origine de l'écho

Le problème de l'écho est primordial dans un serveur vocal. En effet, il ne faut pas que l'ordinateur interprète l'écho du message qu'il diffuse sur la ligne téléphonique comme un ordre en provenance du locuteur.

Théoriquement, si l'on utilise des lignes numériques sur tout le trajet entre l'interlocuteur et le serveur, il n'y a pas d'écho possible. Par contre, si le locuteur possède une ligne analogique, la non adaptation parfaite du téléphone à la ligne risque de générer un écho (figure 3). Celui-ci est numérisé dans le signal digital au central téléphonique public. Il est ainsi contenu dans le flot de bits qui partent en direction du serveur vocal.

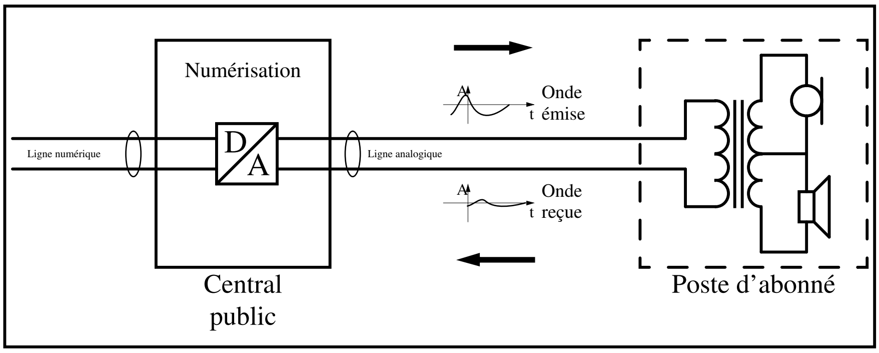


FIGURE 3. Provenance de l'écho

La réflexion est caractérisée par le coefficient de réflexion $\underline{\rho}(x)$, qui est le rapport (complexe) entre la tension $\underline{U}_r$ associée à l'onde rétrograde et la tension $\underline{U}_i$ associée à l'onde incidente.

$$\underline{\rho}(x) = \frac{\underline{U}_r}{\underline{U}_i}$$

En définissant $\underline{Z}_c$ comme étant l'impédance caractéristique de la ligne, et $\underline{Z}_L$ la charge connectée à l'extrémité de cette ligne, on peut réécrire l'équation ci-dessus de la manière suivante:

$$\underline{\rho}(x) = \frac{\underline{Z}_L - \underline{Z}_c}{\underline{Z}_L + \underline{Z}_c}$$

Trois cas intéressants peuvent être extraits de cette formule:

- la ligne est chargée par son impédance caractéristique $\underline{Z}_L = \underline{Z}_c$. La formule ci-dessus donne $\underline{\rho}(x) = 0$. Il n'y a donc aucune réflexion, raison pour laquelle la ligne est dite "adaptée". La charge absorbe toute l'énergie transmise.

- la ligne est terminée par une charge inférieure à son impédance caractéristique $\underline{Z}_L < \underline{Z}_c$. On a alors $\underline{\rho}(x) < 0$. Cela signifie que l'onde réfléchie est en opposition de phase avec l'onde incidente. Dans le cas extrême d'un court-circuit, $\underline{\rho}(x) = -1$.
- la ligne est terminée par une charge supérieure à son impédance caractéristique $\underline{Z}_L > \underline{Z}_c$. On a alors $\underline{\rho}(x) > 0$. Cela signifie que l'onde réfléchie est en phase avec l'onde incidente. Dans le cas extrême du circuit ouvert, $\underline{\rho}(x) = 1$.

De plus, le temps de propagation du signal sur le media de transmission retarde le signal lors de son parcours de l'émetteur vers le récepteur. En effet, le signal progresse sur la ligne à une vitesse finie. Il faut donc un certain temps pour que le signal atteigne l'extrémité de la ligne et, en cas de non adaptation parfaite, fasse le chemin retour.

6. Un mot sur le projet de semestre

Pour pouvoir entreprendre des recherches dans la direction de l'annulation d'écho, il me faut des outils qui me permettent de traiter des flots de données sonores. C'est le travail effectué lors du projet de semestre qui m'a permis de créer ces outils. Les quelques paragraphes suivants présentent les différentes étapes de la préparation du travail de diplôme effectuées lors du travail de semestre.

6.1 Implémentation full-duplex

Le premier but est de me familiariser avec l'environnement de développement d'applications vocales SunXTL de SunSoft. Cet environnement fournit toute une série de classe C++ de relativement bas niveau qui permettent de gérer différents types de media. Toutes ces classes offrent au programmeur la possibilité de créer des applications vocales indépendantes du hardware de la machine utilisée (figure 5).

Dans le but de démontrer les possibilités de l'environnement de développement SunXTL, la société aComm a écrit une petite application de démonstration, dénommée MailVox. Cet exemple permet de simuler un répondeur classique, c'est-à-dire que dans un premier temps, il diffuse un message à l'attention de l'appelant. Puis, dans un deuxième temps, il se met en mode d'enregistrement et stocke le message de l'appelant sur le disque dur ou tout autre périphérique connecté à l'ordinateur.

La modification du code de cet exemple doit permettre de lancer simultanément une opération de diffusion et une opération d'enregistrement (figure 4). Ces deux tâches doivent s'appliquer au même flux de données et ne pas se gêner ou s'interrompre l'une l'autre.

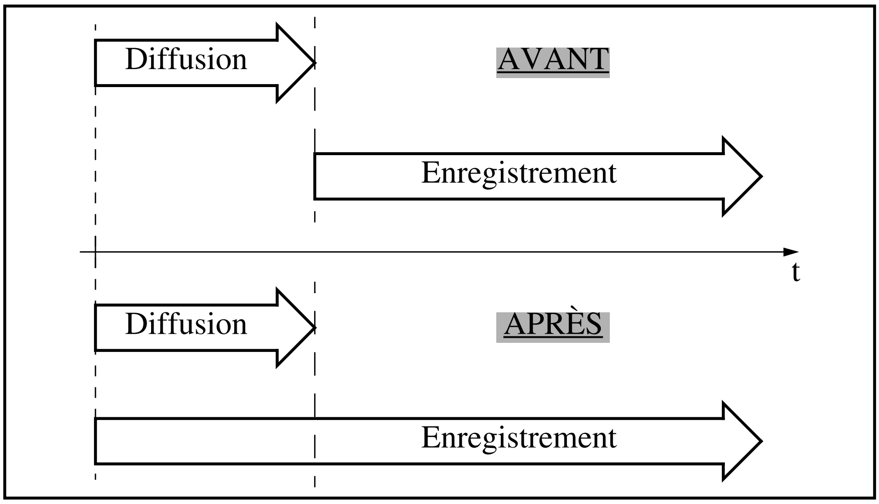


FIGURE 4. Modification full-duplex

Les différentes modifications du code sont présentées dans le paragraphe "Adaptations du logiciel" en page 11.

6.2 SunXTL en bref

Le design et l'architecture de SunXTL sont prévus pour atteindre plusieurs buts principaux *[12]*:

- Fournir une API (*Application Programming Interface*) au programmeur pour le développement d'applications personnelles.
- Fournir une portabilité transparente entre les technologies analogiques, RNIS, ATM (*Asynchronous Transfer Mode*) et autres. L'application ne doit pas être recompilée lorsque l'on change de media ou de technologie.
- Proposer un accès facile aux services vocaux classiques. Elle fournit par exemple un outil de génération/détection DTMF (*Dual Tone Multi Frequency*).

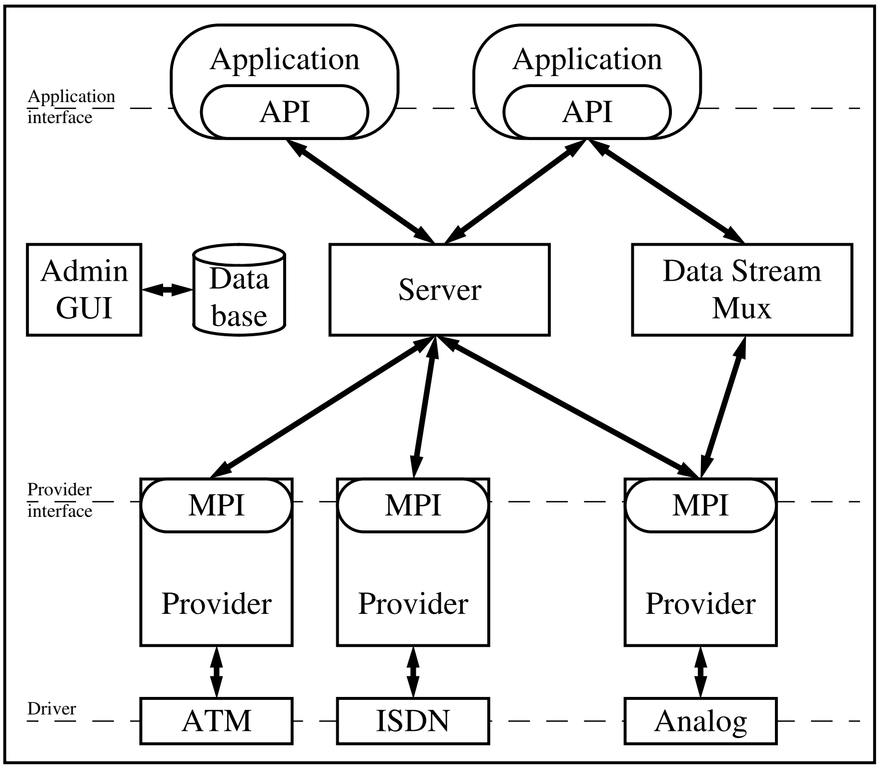

Application
API
Application
API
Application interface
Admin GUI
Data base
Server
Data Stream Mux
Provider interface
MPI
Provider
MPI
Provider
MPI
Provider
Driver
ATM
ISDN
Analog

FIGURE 5. L'environnement SunXTL

L'API SunXTL est une interface orientée objet accessible en C++ *[13]*. Elle met à disposition du programmeur trois types de méthodes: requêtes asynchrones, commandes asynchrones et événements asynchrones. Différentes méthodes permettent la réponse à un appel, le transfert d'appel entre plusieurs applications, l'enregistrement ou la diffusion de messages. D'autres méthodes permettent de contrôler l'état d'un appel en cours et, le cas échéant, de modifier cet état. L'environnement s'appuie, pour la gestion des appels, sur la notion de *Provider* qui est en quelque sorte un driver hardware fourni par le fabricant de la carte utilisée. Les MPI (*Media Provider Interface*) font le lien entre le hardware de la carte d'interface et le software de l'application.

Le système SunXTL se comporte alors comme une couche intermédiaire entre le hardware et l'application. Le serveur SunXTL, de conserve avec les librairies de l'application et du provider, prend en charge le passage de messages entre processus, la création et l'identification des objets, ainsi que l'appartenance d'un appel, la sécurité et la notification des événements asynchrones.

D'autre part, l'environnement met à la disposition du programmeur plusieurs méthodes annexes et classes utilitaires. Elles permettent de traiter des tableaux de bytes, des chaînes de caractères ainsi que des listes chaînées. Deux autres classes, **dpDispatcher** et **dpIOHandler**, servent à traiter les événements produits par les entrées/sorties. A l'aide de tous ces outils, il est plus ou moins aisé de programmer une application vocale.

Toutes ces information donnent un aperçu condensé des possibilités offertes par SunXTL. Tous renseignements supplémentaires pourront être trouvés dans les documents cités dans la bibliographie (*[12]* et *[13]*).

6.3 Adaptations du logiciel

Toutes les modifications que j'ai apporté au logiciel sont au niveau de l'API. En effet, le code source de tout ce qui est situé en dessous de l'API n'est pas en ma possession. Il n'est donc possible d'apporter des corrections qu'au niveau de l'application MailVox utilisant SunXTL.

6.3.1 Faculté de full-duplex

L'application de démonstration MailVox permet de simuler le comportement d'un répondeur téléphonique classique. C'est-à-dire qu'elle permet de diffuser un message puis, ensuite, d'enregistrer ce que l'interlocuteur dit.

Dans le cadre des serveurs vocaux interactifs, il faut autoriser l'interlocuteur à couper la parole au serveur vocal. Pour ce faire, il faut que l'enregistrement débute simultanément avec la diffusion du message. Le fichier généré par l'enregistrement est scanné en temps réel par un module de reconnaissance vocale qui, dès qu'il détecte un mot clé autorisé, génère un signal pour l'application MailVox. Ce signal est utilisé pour interrompre la diffusion et l'enregistrement en cours, et lancer la diffusion d'un nouveau message adapté à la requête de l'interlocuteur. De plus, il lance l'enregistrement d'un nouveau fichier pour permettre la reconnaissance de l'ordre suivant.

La plus importante modification du code se situe au niveau de la création et de la destruction des objets de gestion des flots d'entrée/sortie.

Avant cette modification, deux objets distincts étaient créés. L'un pour supporter le flot de données provenant de l'interface RNIS, l'autre pour prendre en charge les données destinées à cette même interface. Ces deux objets, du fait des contrôle effectués par le programme, ne pouvaient pas coexister. En effet, seul un objet à la fois peut accéder à l'interface RNIS.

Maintenant, il n'y a plus qu'un seul objet qui est créé pour toute la durée d'une communication. Cet objet prend en charge simultanément l'acheminement des données entrantes et sortantes. Ceci se fait en créant simultanément les tâches de diffusion d'annonce et d'enregistrement de message.

6.4 Analyse des solutions

6.4.1 Le full-duplex

Le serveur est capable de diffuser un message et d'enregistrer les informations provenant de la ligne RNIS simultanément.

Ceci a été rendu possible par les retouches apportées au code de MailVox. Les modifications sont les suivantes:

- La création de l'objet supportant un appel à été déplacé dans la méthode `configuration_ind` de la classe MsgCall. Ceci permet à l'objet de rester actif durant toute la durée de l'appel. Il est détruit seulement lorsque l'appel se termine. Cette opération de destruction à été implémentée dans le destructeur de la classe MsgCall.
- Le lancement des deux tâches de lecture et d'écriture sur le media de communication se fait dans la méthode `configuration_ind` de MsgCall. La tâche d'enregistrement à une légère priorité sur celle de diffusion pour s'assurer que l'on ne perd pas un ordre du locuteur.
- La plus grande partie des tests d'accès au media sont devenus caducs du fait que l'objet est créé et détruit en même temps que l'appel.

7. Concepts

Après la mise en place des outils lors du projet de semestre, je vais traiter dans les pages suivantes des différentes idées que je vais utiliser pour atteindre l'objectif du projet de diplôme. Je vais aussi parler des hypothèses de travail retenues ainsi que des critères de choix en faveur de la méthode sélectionnée.

7.1 Hypothèses de travail

Pour débuter le projet, j'ai dû définir un modèle sur lequel je vais baser mon travail. Ce modèle a les caractéristiques suivantes:

- les affectations du signal se situent uniquement au niveau d'un décalage dans le temps et d'une atténuation constante et uniforme sur tout le signal,
- ce qui suppose la prise en compte d'un canal de transmission présentant une fonction de transfert idéale.
- correction statique et uniforme du signal en fonction des paramètres calculé en début de communication.

7.2 Atténuation et décalage dans le temps

Comme expliqué dans le chapitre "Origine de l'écho", lors de la transmission d'un signal sur une ligne analogique, celui-ci subit deux phénomènes. Premièrement l'atténuation qui est une perte d'amplitude du signal reçu par rapport au signal original. Deuxièmement, le décalage dans le temps qui est principalement dû au temps de propagation du signal sur le media de transmission. Cela est bien sur toujours valable dans le cas de signaux numériques.

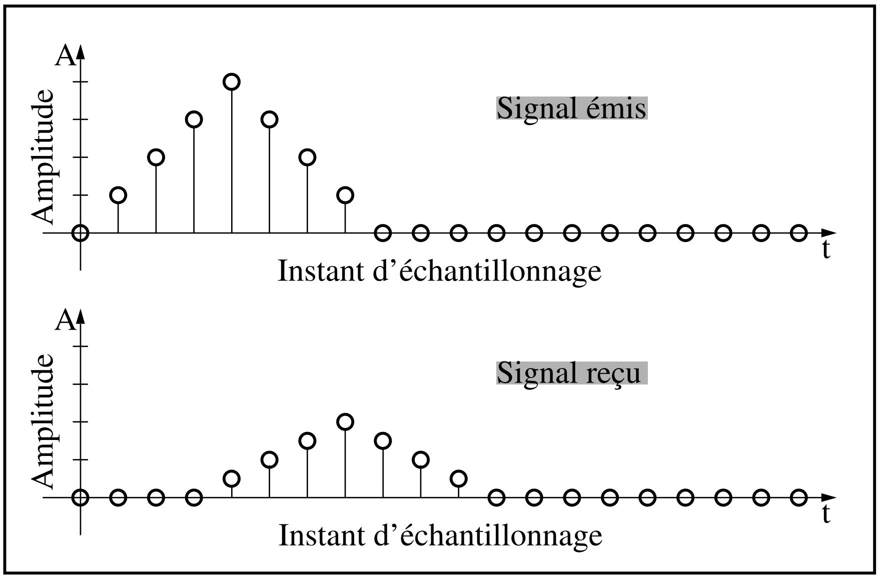


FIGURE 6. Atténuation et décalage dans le temps

Le temps de propagation d'un signal numérique dépend non seulement des caractéristiques propres de la ligne mais aussi de la profondeur des buffers insérés dans le circuit de ligne au central téléphonique. La figure 6 illustre comment un signal émis est modifié par le canal de transmission et la forme sous laquelle il est reçu. Dans cet exemple, le signal reçu est idéal et ne tient pas compte du bruit et des distorsions du signal.

De la même façon, l'écho du signal émis est affaibli et décalé dans le temps puisqu'il parcourt le media de transmission à l'aller et au retour. Il est donc nécessaire de savoir si le signal de retour vers le diffuseur est suffisamment faible pour être ignoré, ou si au contraire, il est assez important pour entraîner des perturbations de fonctionnement du module de reconnaissance vocale.

Par la suite, je vais utiliser le terme de *taux d'écho* que je définis ainsi:

> *Le taux d'écho représente le rapport signé de l'amplitude du signal réfléchi et de l'amplitude du signal émis.*

Si le taux d'écho est important, il est possible que le module de reconnaissance vocale interprète l'écho de son propre message comme un ordre et ainsi qu'il s'auto-commande. Dans ce cas, il est nécessaire de corriger le signal reçu à l'aide d'une méthode d'annulation de l'écho afin de pouvoir correctement distinguer d'une part les ordres de l'utilisateur et d'autre part les messages du serveur.

L'une des idées pour trouver le taux d'écho et le temps de propagation du signal est de générer en début de communication un signal caractéristique d'environ 100ms que l'on va s'efforcer de retrouver dans le signal reçu par le serveur. La méthode communément utilisée pour faire ce travail est la corrélation qui mesure la ressemblance d'un signal par rapport à l'autre pour un décalage relatif donné.
C'est cette méthode que je me propose de mettre en oeuvre pour la réalisation du projet.

L'opération de corrélation peut se réaliser directement sur les signaux dans le domaine du temps ou sur la transformée de Fourier des signaux. Le choix parmi ces deux méthodes est surtout dicté par le temps de calcul nécessaire.

7.3 La corrélation

L'autocorrélation donne une mesure de la ressemblance entre le signal et une version décalée dans le temps du même signal. De manière similaire, on définit la corrélation croisée qui est une mesure de la ressemblance entre un signal et une version décalée d'un autre signal. La fonction de corrélation croisée peut être utilisée pour reconnaître un signal connu dans un signal aléatoire. C'est cette propriété qui est mise à profit dans ce travail pour retrouver le signal émis en début de communication à l'intérieur du signal enregistré par le serveur.

Pour fixer la notation, dans la suite du document, les signaux sont définis comme suit:

- le signal portant l'indice 1 est le signal dans lequel la recherche est effectuée.
- le signal portant l'indice 2 étant quant à lui la séquence que l'on essaie de retrouver.

Sous forme mathématique, la corrélation $R_{12}(\tau)$ de deux signaux $g_1(t)$ et $g_2(t)$ s'écrit de la manière suivante:

$$R_{12}(\tau) = \int_{-\infty}^{\infty} g_1(t) \cdot g_2(t-\tau)dt$$

avec les signaux $g_1(t) = g_2(t)$ pour la fonction d'autocorrélation et $g_1(t) \neq g_2(t)$ pour la fonction de corrélation croisée.

Pour plus de clarté, je définis les deux formules suivantes pour des signaux continus:

$$\text{Autocorrélation : } A_c(\tau) = \int_{-\infty}^{\infty} g_2(t) \cdot g_2(t-\tau)dt$$

$$\text{Corrélation croisée : } C_c(\tau) = \int_{-\infty}^{\infty} g_1(t) \cdot g_2(t-\tau)dt$$

L'autocorrélation et la corrélation croisée se calculent de la façon suivante:

- décalage relatif des signaux l'un par rapport à l'autre de τ.
- multiplication des deux signaux τ au considéré.
- effectuer l'intégrale du produit.

Remarques:

- Même si les signaux sont à puissance moyenne finie, on peut concrètement appliquer la théorie des signaux à énergie finie car les signaux sont en pratique limités dans le temps.
- La corrélation est une opération gourmande en ressources de calculs, mais reste toutefois l'opération de base pour l'identification d'un signal dans un autre. Il faut donc veiller à prendre en compte des signaux de relative courte durée pour minimiser le nombre d'opérations de calcul. Cette notion sera développée au chapitre "Évaluation des méthodes de calcul".

Il faut noter que le résultat $R_{12}(\tau) = R_{21}(-\tau)$, ce qui signifie que $R_{12}(\tau)$ et $R_{21}(\tau)$ sont différents. Il est donc important de ne pas se tromper lors du calcul et de bien savoir quel est le signal recherché et quel est le signal dans lequel la recherche est effectuée.

Lorsque l'on passe dans le domaine des signaux discrets, la méthode reste identique. Seul le calcul de l'intégrale change et revient à sommer les valeurs du produit des échantillons à l'instant donné. La variable libre τ devient elle aussi discrète, raison pour laquelle elle est dénommée Δt. La relation mathématique devient alors:

$$R_{12}(\Delta t) = \sum_n g_1(n) \cdot g_2(n - \Delta t)$$

Comme précédemment, je définis les fonctions d'autocorrélation et de corrélation croisée pour des signaux discrets de la façon suivante:

$$\text{Autocorrélation : } A_d(\Delta t) = \sum_n g_2(n) \cdot g_2(n - \Delta t)$$

$$\text{Corrélation croisée : } C_d(\Delta t) = \sum_n g_1(n) \cdot g_2(n - \Delta t)$$

L'intervalle de sommation est *n*, ce qui signifie simplement que la somme s'applique à tous les échantillons qui composent le signal en cours de traitement, quelle que soit sa taille.

Chaque calcul de la somme donne un seul et unique point de la courbe de corrélation croisée ou d'autocorrélation. Pour obtenir la courbe complète, il faut répéter l'opération pour chaque Δt des signaux, c'est à dire sur le nombre d'échantillons qui composent le signal. Un exemple de calcul d'un point de cette courbe est présenté à la figure 7. Les signaux utilisés sont ceux définis à la figure 6.

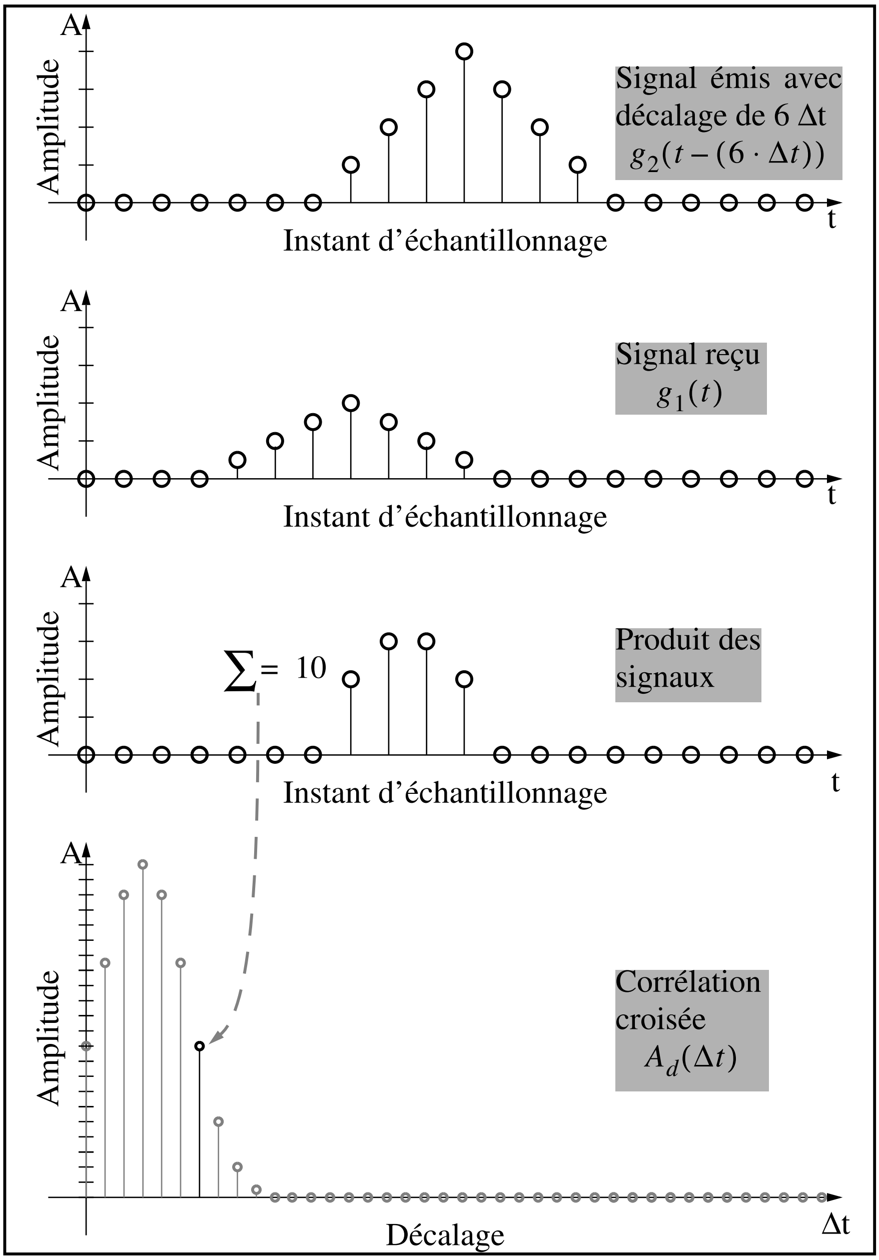

A
Amplitude
Signal émis avec décalage de 6 Δt
$g_2(t-(6\cdot\Delta t))$
t
Instant d'échantillonnage
A
Amplitude
Signal reçu
$g_1(t)$
t
Instant d'échantillonnage
A
Amplitude
$\sum = 10$
Produit des signaux
t
Instant d'échantillonnage
A
Amplitude
Corrélation croisée
$A_d(\Delta t)$
Décalage
Δt

FIGURE 7. Calcul de la corrélation

L'instant pour lequel la fonction de corrélation donne le maximum indique le temps de propagation aller-retour sur le media de transmission.

Dans la formule de la corrélation des signaux continus, si un facteur multiplicatif est attribué à l'un des deux signaux, on peut le mettre en évidence. Or, le calcul de l'autocorrélation et de la corrélation croisée sont

$$\int_{-\infty}^{\infty} a \cdot g_1(t) \cdot g_2(t-\tau)dt = a \cdot \int_{-\infty}^{\infty} g_1(t) \cdot g_2(t-\tau)dt$$

identiques à l'exception précisément du taux d'écho. Si l'on considère ce taux d'écho comme facteur du signal reçu, on peut appliquer la propriété décrite ci-dessus.

Ce raisonnement est aussi valable lorsque l'on traite des signaux discrets. La relation devient alors:

$$\sum_n a \cdot g_1(n) \cdot g_2(n-\Delta t) = a \cdot \sum_n g_1(n) \cdot g_2(n-\Delta t)$$

Donc, pour calculer le taux d'écho, il faut faire le rapport des maxima des fonctions d'autocorrélation et de corrélation croisée. Ce concept est présenté à la figure 8. L'exemple se base sur les signaux définis à la figure 6.

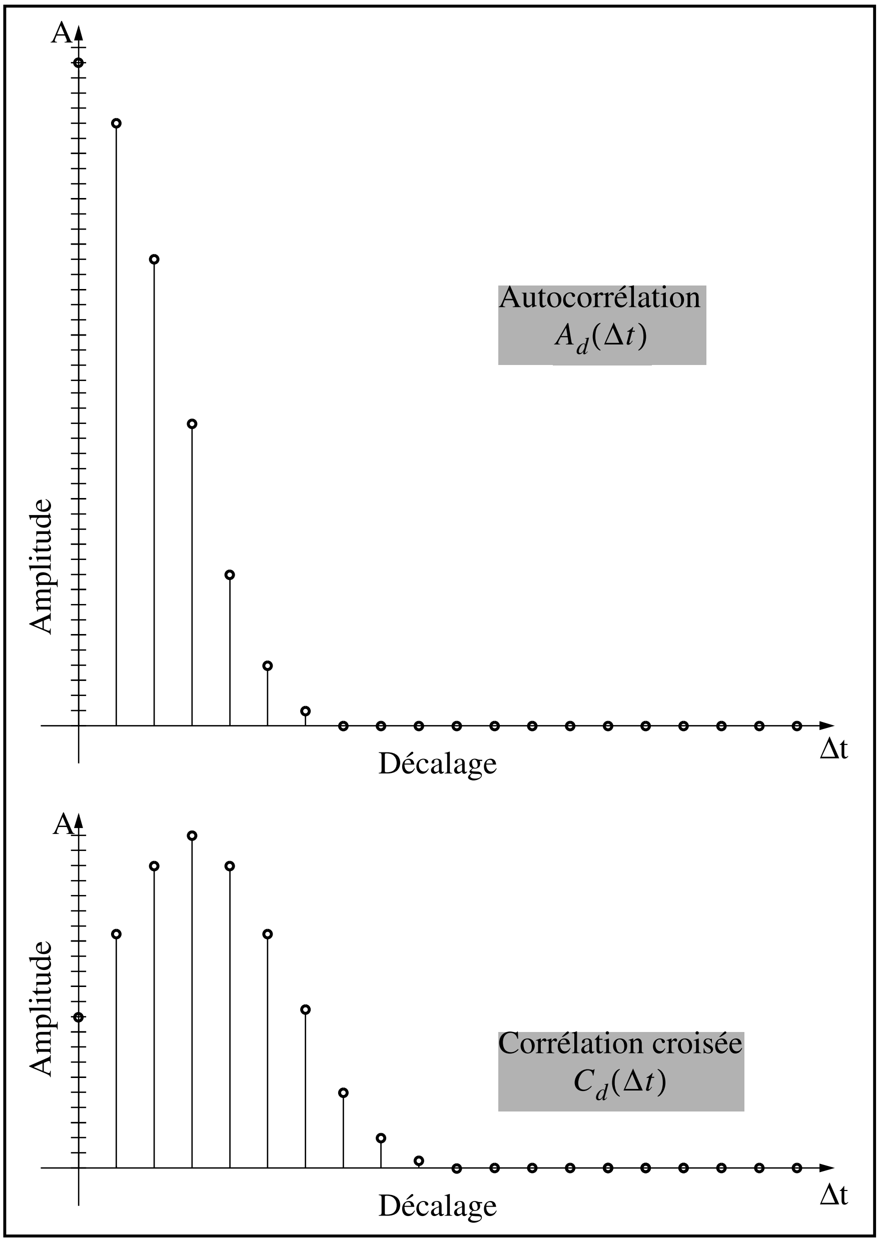


FIGURE 8. Calcul du taux d'écho

Dans cet exemple, la corrélation croisée présente un maximum dont la valeur est la moitié de celle de l'autocorrélation. Cela signifie que le signal reçu est atténué de moitié lors de son parcours aller-retour sur la ligne.

En effet, dans l'exemple donné ci-dessus, la définition des signaux est la suivante: $g_1(t) = 1/2 \cdot g_2(t)$.

L'autocorrélation donne:

$$A_d(\Delta t) = \sum_n g_2(n) \cdot g_2(n - \Delta t)$$

La corrélation croisée donne quant à elle:

$$C_d(\Delta t) = \sum_n g_1(n) \cdot g_2(n - \Delta t) = \frac{1}{2} \cdot \sum_n g_2(n) \cdot g_2(n - \Delta t) = \frac{1}{2} \cdot A_d(\Delta t)$$

La formule permettant de calculer le taux d'écho est:

$$T_e = \frac{max_{sig}(|C_d(\Delta t)|)}{max_{sig}(|A_d(\Delta t)|)}$$

où la fonction max_{sig} retourne le maximum signé de la valeur absolue de la fonction. En effet, il se peut que, contrairement à tous les graphiques présentés jusqu'ici, le maximum en valeur absolue de $A_d(\Delta t)$ ou de $C_d(\Delta t)$ soit négatif.

L'application de la formule du calcul du taux d'écho à l'exemple précédent donne effectivement le taux d'écho de 50% utilisé.

La conversion en décibels se fait à l'aide de la formule suivante:

$$\text{Taux d'écho en [dB]} = 20\log(T_e)$$

Comme présenté dans le chapitre "Origine de l'écho", le signe du rapport des maxima donne une indication sur le coefficient de réflexion ρ de la ligne.

Pour mémoire, trois cas peuvent se présenter:

- si le taux d'écho est positif, le signal réfléchi est ajouté au signal incident en bout de ligne.
- si le taux d'écho est nul, la charge est adaptée à la ligne. Par conséquent, il n'y a pas d'écho.
- si le taux d'écho est négatif, le signal réfléchi est soustrait au signal incident en bout de ligne.

8. Présence de l'écho

La première tâche de mon travail est de mettre en évidence la présence ou non d'un écho dans le signal qui est reçu par le serveur. L'expérimentation m'a permis de distinguer deux cas:

- liaison numérique de bout en bout, sans aucune conversion numérique -> analogique au central téléphonique.
- liaison numérique du serveur au central, puis analogique du central à l'utilisateur, donc avec une conversion numérique -> analogique au central téléphonique.

Pour obtenir une information valable sur le taux d'écho, il est impératif d'avoir les mêmes conditions de test pour les deux cas. Les conditions de test sont les suivantes:

- diffusion d'un message vocal de type extrait de voix.
- déconnexion de l'écouteur de la station téléphonique pour éviter le couplage sonore entre le microphone et le haut-parleur par l'air.

Les deux essais ont confirmés la supposition émise au chapitre "Origine de l'écho" selon laquelle un écho n'apparaît que lors de la liaison analogique.

Le signal émis lors de la liaison analogique est aisément reconnaissable dans le signal enregistré par le serveur. Cela montre donc la nécessité de la correction du signal comme envisagée et présentée dans le chapitre des concepts. Je tiens à réitérer ici l'hypothèse de travail retenue, à savoir une correction statique et uniforme du signal en fonction des paramètres calculés grâce à la corrélation.

La non adaptation de l'appareil téléphonique à la ligne analogique est aussi clairement la source d'écho la plus importante que le signal rencontre au long de son parcours sur la ligne.

Il est quand même intéressant de noter qu'on peut aussi percevoir des atténuations fréquentielles du signal. Cela indique clairement que le canal de transmission n'est pas uniforme dans la bande de fréquence téléphoniques 300-3400Hz.

Cette constatation montre que, dans un premier temps, il serait utile de corriger le signal dans le domaine des fréquences, par exemple en appliquant une égalisation. Ensuite seulement, appliquer l'annulation de l'écho.

9. Signaux utilisés

Le choix de la forme d'onde émise en début de communication est primordiale. C'est en effet sur elle que repose le calibrage de la communication. Pour cette raison, le signal utilisé devrait répondre à trois critères importants:

- il faut qu'il ait une fonction d'autocorrélation qui présente un maximum franc et unique.
- la dynamique du signal doit être importante afin que son écho se distingue aisément par rapport au bruit.
- le spectre fréquentiel du signal doit se situer dans la bande passante téléphonique normalisée de 300 à 3400 Hz.

Accessoirement, il faudrait que le signal ne soit pas désagréable à entendre pour l'utilisateur.

Toutes ces considérations me permettent d'éliminer rapidement les signaux à caractère périodique. En effet, à chaque demi-période, la fonction d'autocorrélation croît, et ce linéairement. On obtient un résultat relativement triangulaire qui ne présente pas la netteté du maximum requise. C'est pour cette raison que l'utilisation de signaux périodiques n'est pas vraiment adaptée.

Cela justifie ma motivation à utiliser des signaux non périodiques tels que la fonction sinus cardinal (figure 10) ou directement un extrait de voix (figure 9).

Les caractéristiques intéressantes de ces deux types de signaux sont présentées dans les deux prochains paragraphes.

9.1 Extrait de voix

Ce type de signal présente un avantage important par rapport à tous les autres signaux utilisables, c'est le fait qu'il fait partie intégrante de l'information que le serveur délivre à l'utilisateur. Par conséquent, ce dernier n'a pas “conscience” du calibrage de la ligne puisqu'aucun signal caractéristique autre que le signal d'information est diffusé sur la ligne.

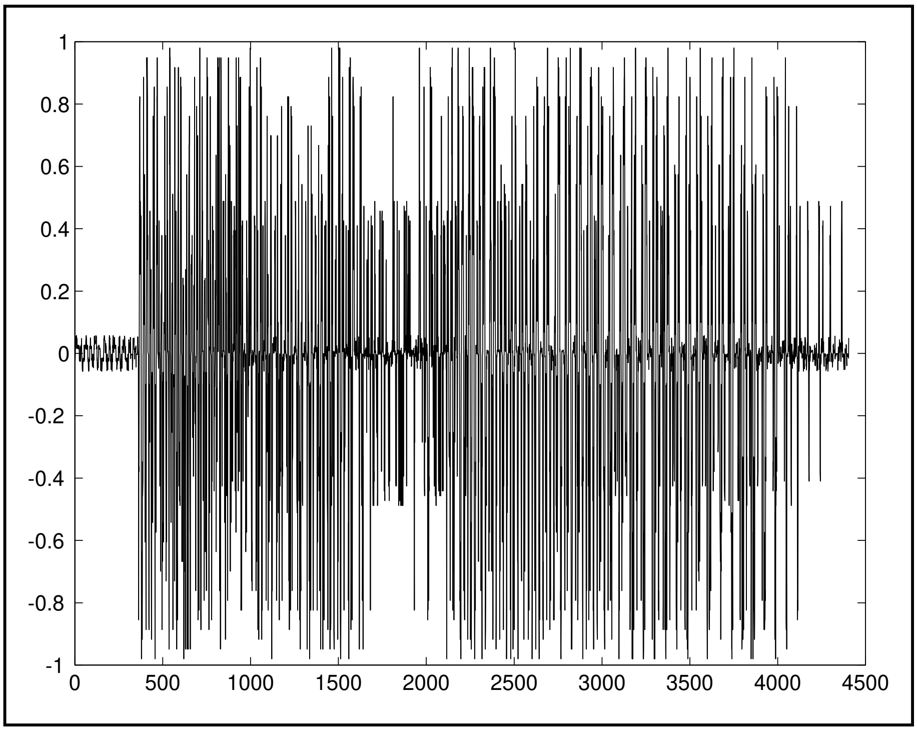


FIGURE 9. Extrait de voix

Il a aussi l'avantage de ne pas présenter de périodicité marquée et de se situer par nature dans la bande de fréquence téléphonique.

Par contre, avec un tel signal, on ne peut pas garantir que le maximum de la fonction d'autocorrélation soit unique. Cela pose des problèmes pour la recherche de ce maximum et par conséquent pour le calcul du retard de l'écho. Il est difficilement applicable dans le cadre du calcul des paramètres de calibrage.

9.2 La fonction sinus cardinal

La fonction sinus cardinal (figure 10) est définie mathématiquement par la relation suivante:

$$y = \frac{\sin(\pi x)}{\pi x} = \operatorname{sinc}(x)$$

Elle est intéressante à plusieurs titres. Premièrement, elle à une transformée de Fourier qui couvre un spectre de fréquences rectangulaire. Deuxièmement, elle présente un maximum important dans sa fonction d'autocorrélation. Finalement, elle n'est pas périodique.
Elle répond donc à tous les critères de choix que j'ai définis auparavant. C'est pour cette raison que je vais concentrer mon travail autour de l'utilisation de ce signal particulier.

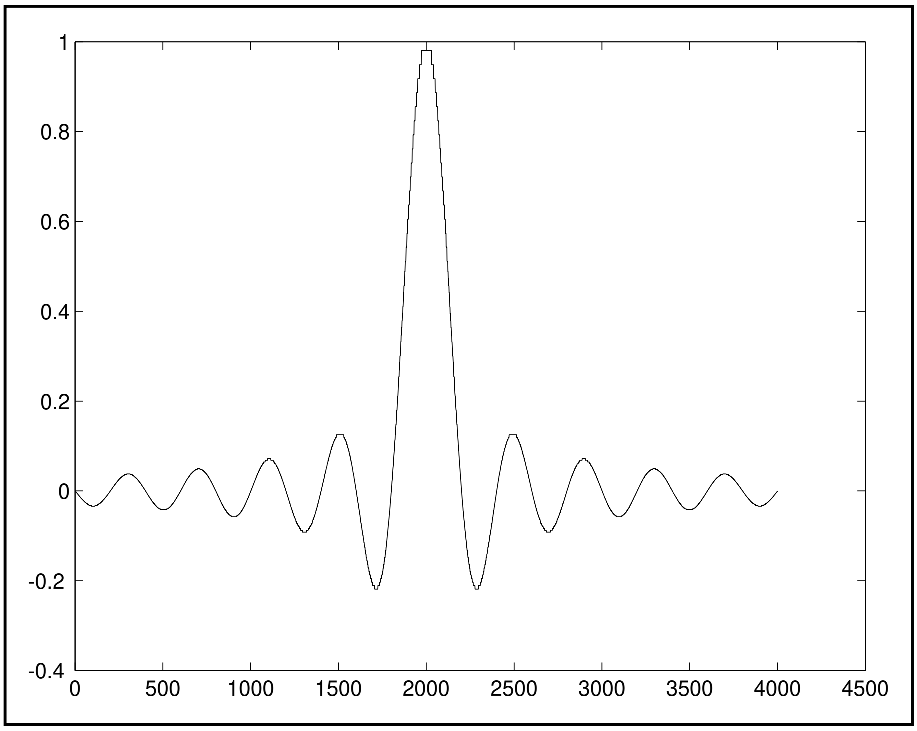


FIGURE 10. Fonction sinus cardinal

10. Évaluation des méthodes de calcul

La seconde étape du travail doit permettre de sélectionner la méthode de calcul la plus efficace pour la corrélation. Pour ce faire, j'ai pris en considération les deux méthodes communément utilisées, à savoir la corrélation directement dans le domaine du temps et la corrélation avec passage dans le domaine des fréquences. Pour cette dernière, une bonne connaissance de la théorie de Fourier est indispensable.
Voici donc expliqués ci-après les concepts utilisés pour le calcul de la corrélation dans le domaine des fréquences.

La transformée de Fourier est une opération qui permet d'analyser un signal et d'en extraire l'amplitude et la phase de ses composantes fréquentielles. Mathématiquement, elle est définie ainsi:

$$G(f) = \int_{-\infty}^{\infty} g(t) \cdot e^{(-j2\pi ft)} dt$$

où $j = \sqrt{-1}$, c'est à dire le nombre imaginaire.

Cette transformation convertit la représentation temporelle d'un signal en représentation fréquentielle. Les domaines du temps et des fréquences sont donc liés. La transformation de Fourier a plusieurs propriétés intéressantes dont celle-ci particulièrement utile dans le cadre de ce travail:

$$g_1(t)*g_2(t) \Leftrightarrow G_1(f) \cdot G_2(f)^*$$

Cette relation est connue sous le nom de théorème de la convolution. Elle est symbolisée par l'opérateur *. Elle revient à dire que l'opération de convolution dans le domaine temporel se traduit dans le domaine fréquentiel par la multiplication de la transformée de Fourier du signal $g_1(t)$ avec le conjugué complexe de la transformée de Fourier du signal $g_2(t)$ *[3]*.

La différence entre la convolution et la corrélation est minime. Le produit de convolution est la fonction de corrélation de deux signaux dont l'un est rabattu autour de l'axe des ordonnées. On voit donc qu'il est aisé d'utiliser cette propriété pour le calcul de la corrélation *[2]*.

Les questions suivantes se posent alors:

- est-ce que la transformation de Fourier fait gagner du temps de calcul sur l'opération de corrélation?
- comment appliquer la transformée de Fourier à un signal échantillonné?

10.1 DFT et FFT

La DFT (Discrete Fourier Transform) est le correspondant discret de la transformée de Fourier continue.

$$G(k) = \sum_{k=0}^{N-1} g(k) \cdot e^{\left(-j2\pi\frac{k}{N}\right)} \qquad (0 \le j \le N-1)$$

Elle fait correspondre une séquence périodique discrète $g(k)$ (où k est entier) de période N, à une autre séquence périodique $G(k)$ de coefficients fréquentiels. La transformée de Fourier, inverse aussi bien que directe, d'une fonction échantillonnée est une fonction périodique dont la période est l'inverse de la période d'échantillonnage. Dans le cas de l'échantillonnage pratiqué sur les liaisons téléphoniques numériques, la période d'échantillonnage est de $125\mu s$.

Le nombre d'opérations de multiplication requises pour exécuter le calcul de la DFT est proportionnel à N^2 *[2]*. C'est à cela qu'il faut comparer les $N \cdot \log N$ opérations nécessaires au calcul de la FFT (Fast Fourier Transform) (voir figure 11).

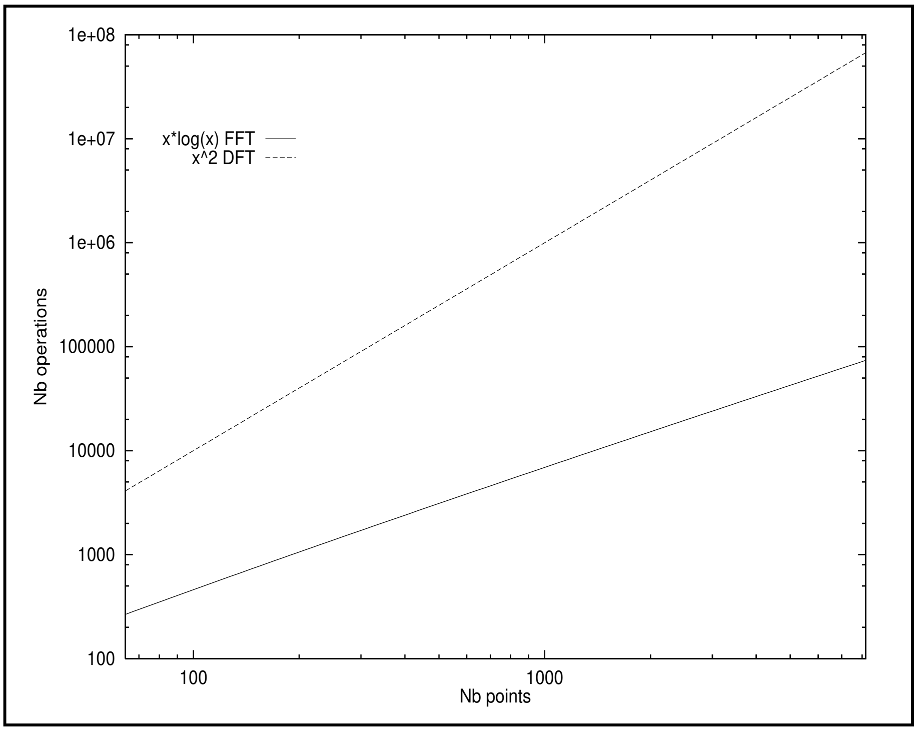


FIGURE 11. Comparaison du nombre d'opérations pour la DFT et la FFT

La FFT est un algorithme de calcul qui tire parti des propriétés de symétrie des racines complexes. La contrainte pour pouvoir appliquer la FFT est que le nombre d'échantillons de la séquence doit être une puissance de 2. Si ce n'est pas le cas, on peut allonger la séquence avec des zéros ou couper celle-ci. Il est en effet beaucoup plus rapide de calculer la FFT d'une séquence de 4095 échantillons sur 4096 points qui est une puissance de 2. À titre d'exemple, The Mathworks Inc., le créateur du logiciel MatLab, donne les temps suivants:
Une machine sur laquelle le temps de calcul d'une FFT sur 4096 points requiert 2,1 secondes, le calcul de la FFT sur 4095 et 4097 points demande respectivement 7 et 58 secondes.

Afin d'avoir une idée de la différence de temps de calcul dans les domaines temporels et fréquentiels, j'ai simulé ces deux opérations à l'aide du logiciel MatLab. J'ai utilisé pour cela la fonction sinc (sinus cardinal).

10.2 Corrélation dans le temps

Le script MatLab *sansfft.m*, fourni sur la disquette d'annexes, donne la méthode utilisée pour la vérification du concept de corrélation dans le domaine du temps.

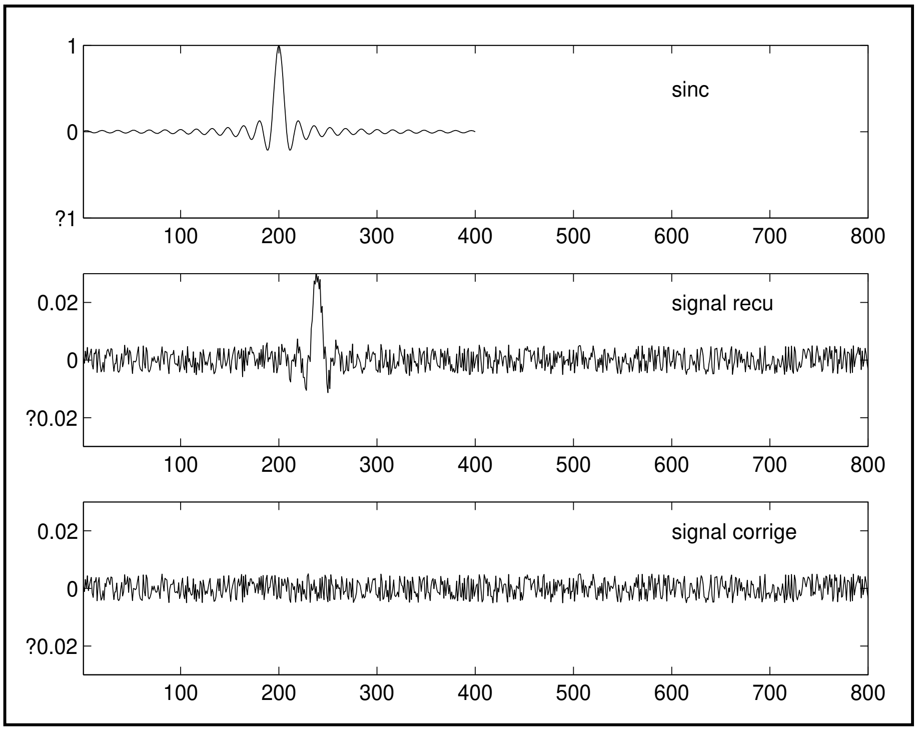


FIGURE 12. Résultat de la simulation dans le domaine du temps

La figure 12 présente les résultats de cette simulation sous forme graphique.

Le premier graphe représente l'impulsion sinc originale. Le signal reçu est calculé en fonction des paramètres de taux d'écho et de délai déterminés dans le script, avec de plus, l'ajout d'un signal aléatoire de bruit. Le dernier graphe présente l'allure du signal reçu modifié par le signal de correction calculé.

On s'aperçoit rapidement que l'écho du signal émis est noyé dans le bruit additionné au signal original. Cela ne veut pas forcément dire que le signal d'écho à été complètement et parfaitement corrigé, mais la correction apportée permet de rendre invisible l'écho vis à vis du bruit.

Ce premier résultat m'indique que la méthode choisie pour réaliser l'annulation de l'écho est envisageable. Il me faut encore tester l'autre méthode qui utilise la FFT.

10.3 Corrélation dans la fréquence

La seconde possibilité de réaliser la corrélation est celle qui consiste à multiplier la transformée de Fourier du signal $g_1(t)$ avec le conjugué complexe de la transformée de Fourier du signal $g_2(t)$. Là aussi, le script MatLab *avecfft.m*, fourni sur la disquette d'annexes, m'a permis d'expérimenter cette méthode.

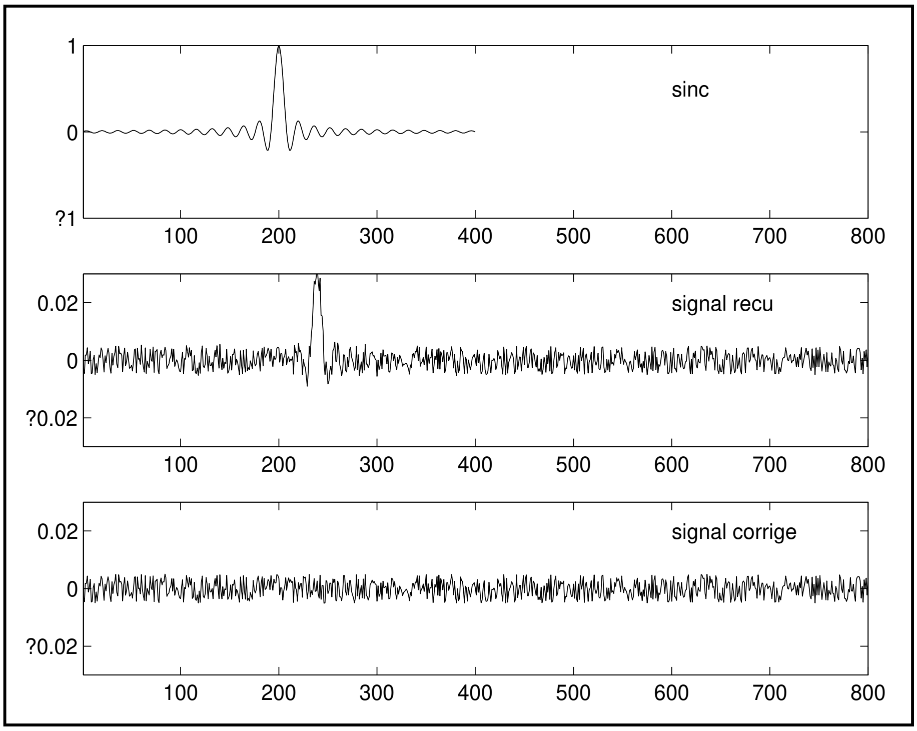


FIGURE 13. Résultat de la simulation dans le domaine des fréquences

Comme pour l'opération dans le domaine du temps, le signal reçu est calculé en fonction du signal émis et des paramètres d'écho et de délai. Et là encore, on observe clairement que le signal corrigé est entièrement noyé dans le bruit.

La figure 13 présente les résultats sous la forme graphique obtenus lors de la simulation. Les résultats sont similaires à ceux obtenus précédemment à la figure 12. Heureusement, puisque les deux méthodes doivent donner les mêmes résultats.

Ces deux expérimentations similaires avec l'outil MatLab m'amènent aux réflexions suivantes:

- la technique retenue pour effectuer l'annulation de l'écho est théoriquement réalisable, simulations à l'appui.
- les deux méthodes, temporelles ou fréquentielles, donnent des résultats identiques.

Il me faut maintenant déterminer laquelle des deux approches est la plus efficace, que ce soit en termes de mise en oeuvre au niveau du logiciel que je dois écrire, ou en termes de performances.

Le premier point est relativement simple à traiter. En effet, dans les deux cas, l'algorithme est clairement défini. De plus, de nombreux *packages* mathématiques disponibles en source fournissent les routines pour faire ces travaux *[16] [17]*. Il m'est donc possible de m'inspirer de ce qui existe déjà afin de gagner en efficacité et en temps.

Pour le second point par contre, seule une expérimentation sur les temps de calcul respectifs des deux méthodes peut m'apporter une réponse. J'ai donc entrepris de simuler encore une fois à l'aide de MatLab le comportement de chacun des algorithmes. J'ai pour cela écrit un script, *comptest.m* fourni sur la disquette d'annexes, dans lequel je fais faire un certain nombre de fois la même opération pour chaque algorithme. À chaque opération, je mesure le temps requis pour le traitement. Le nombre de points sur lequel l'opération s'effectue augmente à chaque itération, allant de 2^6 points à 2^{13} points par puissance de 2. Le pourquoi du nombre de points égal à une puissance de deux est donné par les contraintes de l'algorithme de la FFT qui n'est réellement efficace que si $\log_2(\text{Nb points}) \in N$ (ensemble des nombres naturels).

Les résultats de cette simulation sont donnés sous forme graphique à la figure 14. L'analyse de ce graphe confirme le fait que le calcul par la méthode FFT est beaucoup plus rapide que le calcul par la méthode temporelle. La comparaison avec la figure 11 permet aussi de tirer un parallèle entre la corrélation temporelle est la corrélation avec la DFT. Cette dernière n'apporte pas de gain de temps de calcul. Seule la FFT est vraiment efficace pour ce genre de calculs. C'est pourquoi, je vais me concentrer par la suite sur l'écriture d'un programme utilisant l'algorithme de la FFT.

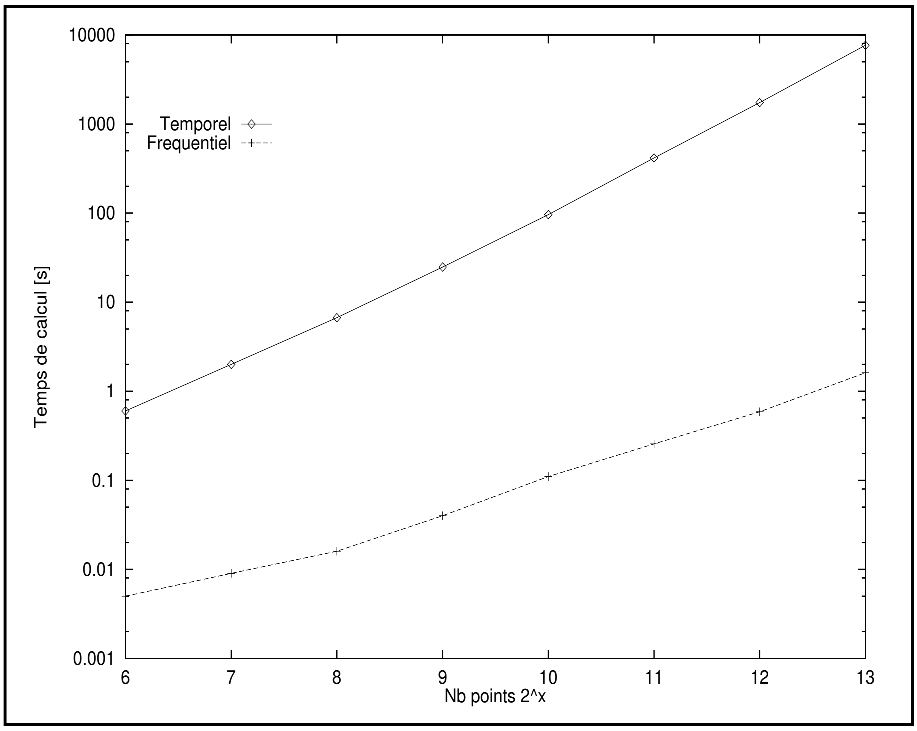


FIGURE 14. Comparaison des temps de calcul

10.4 Résultats

Je vais résumer les informations obtenues à propos des méthodes de calcul:

- Les deux méthodes sont envisageables pour un calcul sur un nombre de points restreint.
- L'option d'utilisation de la FFT s'avère intéressante au niveau du temps de calcul.
- Le gain sur le nombre d'opérations est environ de 2 ordres de magnitude (facteur 100).

Toutes ces données m'ont permis acquérir la certitude que l'utilisation de la FFT est incontournable dans le carde de ce projet. C'est donc cette technique que je vais m'efforcer de mettre en oeuvre dans les prochains chapitres.

En effet, pour un calcul sur un 2048 points équivalent à un signal d'une durée de 256 *ms*, le temps de calcul requis est de 0.11 seconde en utilisant la FFT, et de 962 secondes sans utiliser la FFT. On voit clairement que pour l'application en question, il est indispensable d'utiliser la FFT.

11. Écriture du programme

11.1 Découpage du logiciel

L'utilisation de C++ comme langage de programmation permet l'utilisation de la notion d'objets et l'utilisation des classes. Cela me permet de découper mon programme en modules qui ont chacun une fonction particulière. Cela me donne aussi la possibilité d'insérer dans le logiciel des morceaux de codes déjà écrits et longuement testés.

Plusieurs avantages sont à tirer de cette situation:

- le code a déjà une certaine robustesse.
- gain de temps sur l'écriture du programme et des tests de celui-ci.
- mise en application du principe de réutilisabilité d'une classe écrite en langage orienté objet, en l'occurrence le C++.

Ces quelques remarques préliminaires m'amènent à une brève description du déroulement temporel des diverses opérations à réaliser.
Lors de l'établissement de la communication, il faut extraire les données du fichier d'enregistrement, effectuer le calcul des paramètres d'annulation et corriger le signal reçu. Le calcul des paramètres est implémenté par l'opération de FFT. La figure 15 illustre ce déroulement.

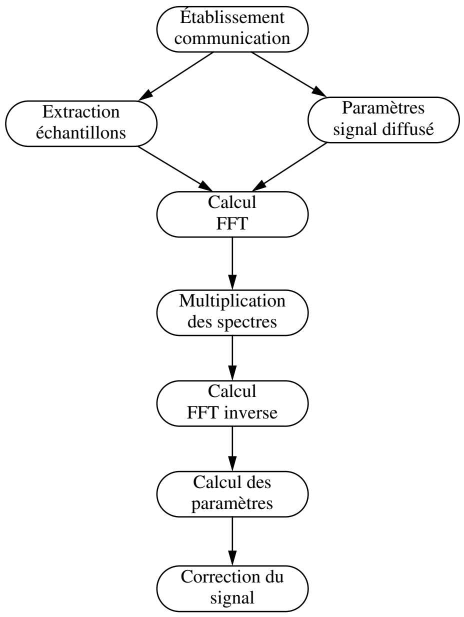

Établissement communication
Extraction échantillons
Paramètres signal diffusé
Calcul FFT
Multiplication des spectres
Calcul FFT inverse
Calcul des paramètres
Correction du signal

FIGURE 15. Déroulement temporel du processus de correction

L'obtention des paramètres du signal diffusé peut se faire de 2 manières différentes:

- calcul de la FFT à chaque exécution du programme
- enregistrement d'une base de données contenant les FFTs des différents signaux utilisés en diffusion.

Dans un premier temps, il a été convenu avec M. Olivier Bornet de l'IDIAP d'écrire un programme indépendant de toute l'application de serveur vocal développé au sein de l'institut. Par contre, le programme doit présenter toute la souplesse voulue pour pouvoir être facilement intégré ultérieurement dans le serveur. Par ailleurs, il me faut aussi fournir la possibilité de traiter des données provenant d'un autre module de l'application.

Toutes ces remarques préliminaires me permettent de présenter le découpage du programme que j'ai retenu pour mettre en oeuvre ce logiciel d'annulation d'écho.

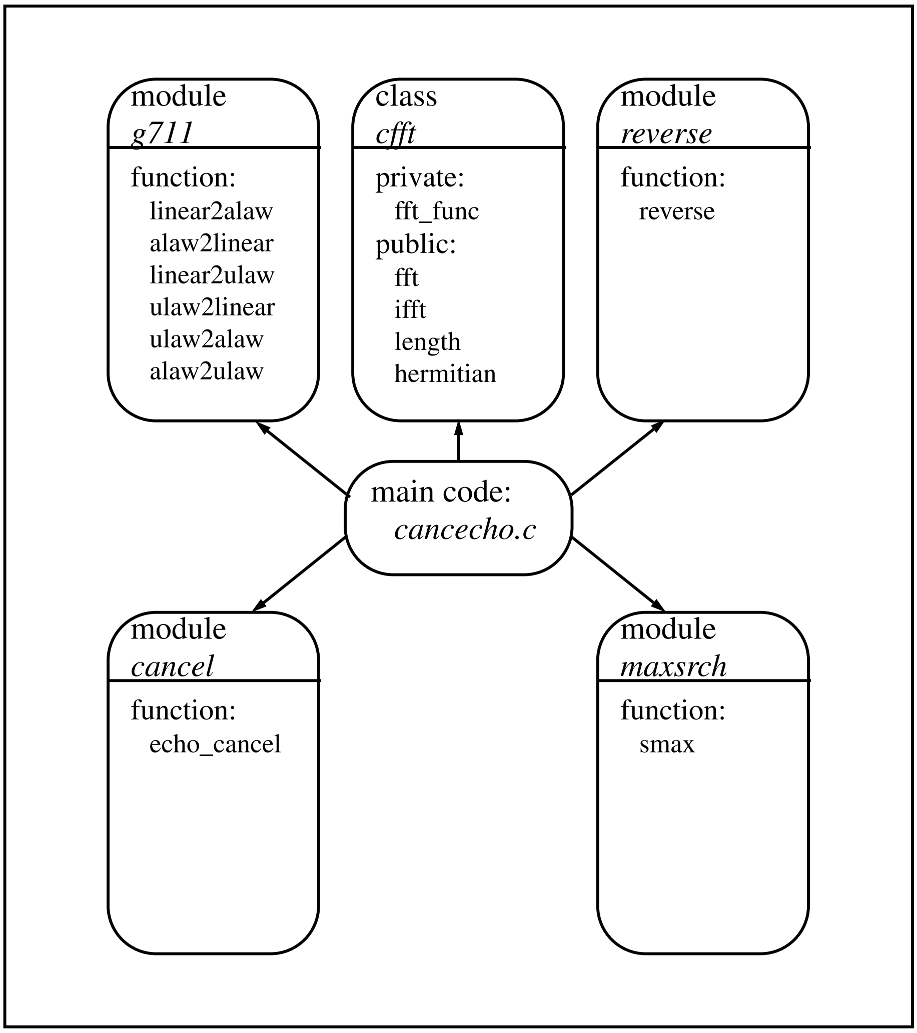

module
g711
function:
linear2alaw
alaw2linear
linear2ulaw
ulaw2linear
ulaw2alaw
alaw2ulaw
class
cfft
private:
fft_func
public:
fft
ifft
length
hermitian
module
reverse
function:
reverse
main code:
cancecho.c
module
cancel
function:
echo_cancel
module
maxsrch
function:
smax

FIGURE 16. Découpage du logiciel

11.2 La classe *cfft*

C'est la classe autour de laquelle tout le programme s'articule. Je ne l'ai pas écrite moi-même, mais je l'ai trouvée sur Internet grâce au forum de news *comp.speech [15]*. Ce forum parle du traitement de la parole assisté par ordinateur. L'un des participants demandait comment implémenter la FFT sur un DSP (Digital Signal Processor) et surtout quel algorithme utiliser. C'est en suivant les liens proposés dans les réponses que j'ai obtenu la classe FFT que j'utilise dans ce programme. L'adresse Web où se trouve le code source est donnée dans la bibliographie *[16]*.

Cette classe fournit les primitives de base pour le calcul de la transformée de Fourier, qu'elle soit directe ou inverse, ainsi qu'une méthode pour obtenir la taille de la séquence de la FFT.
Cette classe est une classe générique. Ceci se distingue par le mot clé *template* qui précède la déclaration de la classe. Ce principe permet d'utiliser la classe avec différents types de données, définis seulement au moment de l'utilisation de celle-ci. Dans ce cas, la classe générique requiert des types de données de forme complexe, c'est à dire que le type doit contenir une partie réelle et une partie imaginaire.

La méthode `fft(CPLX *buf)` exécute le calcul de la FFT sur le tableau de nombres complexes passé en paramètre. La taille du tableau doit être une puissance de deux pour que l'algorithme soit efficace.

La méthode `ifft(CPLX *buf)` exécute le calcul de la FFT inverse sur le tableau de nombres complexes passé en paramètre. Là aussi, la taille du tableau doit être une puissance de deux pour que l'algorithme soit efficace.

La méthode `lenght()` retourne simplement la taille de l'instance courante de la classe *cfft*.

La méthode `hermitian(CPLX *buf)` est utilisée pour remplir la deuxième moitié du spectre complexe d'un signal réel lorsque la première partie est remplie. Elle recopie simplement les coefficients des N/2 premiers éléments du tableau dans les N/2 derniers éléments. Cette méthode n'est pas utilisée dans le cadre de ce projet.

Toutes les méthodes décrites ci-dessus sont publiques, seule la méthode `fft_func(CPLX *buf, int iflag)` est privée. Elle est appelée par les méthodes `fft(CPLX *buf)` et `ifft(CPLX *buf)` pour effectuer le calcul de la FFT. Selon que la valeur du paramètre `iflag` soit 0 ou 1, elle calcule respectivement la transformée de Fourier ou la transformée de Fourier inverse.

Dans le cadre du projet, je vais toujours utiliser des signaux purement réels. C'est à dire que le tableau de complexes associé à une séquence sonore contient le signal dans la partie réelle d'un élément du tableau, et que la partie imaginaire est toujours nulle. Par contre, la transformée de Fourier d'un signal, même purement réel, est toujours complexe. Il faut donc tenir compte dans ce cas de la partie réelle et de la partie imaginaire d'un élément du tableau.

Le code *cplxftt.h* complet avec les commentaires est fourni sur la disquette d'annexes.

11.3 Les modules

11.3.1 Fonctions de conversion

Les fonctions de conversion sont basées sur le code C *g711.c* de Sun Microsystems Inc. qui implémente la norme G711 de l'ITU (International Telecommunication Union) concernant la compression et décompression des échantillons PCM (Pulse Code Modulation) (figure 17) selon les lois A et *[1]*.

Afin de pouvoir utiliser les fonctions implémentées dans *g711.c*, et surtout pouvoir les inclure au code du programme principal, j'ai écrit un fichier d'en-tête nommé *g711.h*, qui contient les déclarations des différentes fonctions du module *g711*.

Il est important de signaler que l'algorithme de la FFT tel qu'il est implémenté dans la classe *cfft* ne permet de travailler que sur des échantillons codés linéairement. Malheureusement, les échantillons provenant d'une ligne RNIS sont codés selon la loi A. Il me faut par conséquent avoir à disposition une fonction qui me permet de décompresser les échantillons pour pouvoir effectuer un calcul correct et une autre me permettant d'effectuer la compression.

Une remarque importante est à donner à propos des algorithmes de compression et de décompression implémentés par les fonctions de *g711.c*.
Le code binaire utilisé pour le codage des échantillons compressé est le code binaire replié avec inversion des bits de rang pair.
En effet, en téléphonie, les signaux de faible amplitude sont les plus fréquents (statistiquement parlant). En code binaire replié simple, cela conduirait au codage de mots PCM composés quasi exclusivement de zéros, ce qui serait très défavorable pour la synchronisation du récepteur et des régénérateurs intermédiaires. C'est pourquoi on inverse les bits de rang pair dans le mot PCM avant de l'émettre, ce qui augmente la probabilité de transition de 1 à 0 et inversement à l'intérieur de chaque mot.
Ceci explique l'opération XOR avec la valeur hexadécimale 0x55 au début de la fonction `alaw2linear`.

D'autre part, ces fonctions utilisent les 12 bits de poids fort d'un entier de 16 bits pour stocker la valeur décompressée de l'échantillon 8 bits. Si l'on considère le codage uniquement sur 12 bits, cela nous donne $2^{12} = 4096$ valeurs avec le bit de signe. Cela donne une gamme de valeurs comprises entre -2048 et +2048. Or les algorithmes retournent des valeurs comprises entre -32768 et +32768, ce qui correspond à un codage sur 16 bits. Pour obtenir la même gamme de valeurs que précédemment, il faut appliquer à l'échantillon décompressé un décalage de 4 bits à droite. Cela n'a pas d'importance pour le calcul car le facteur 16 induit par ce décalage est présent dans les deux calculs des fonctions d'autocorrélation et de corrélation croisée. En appliquant la formule de calcul du taux d'écho, ce facteur apparaît au numérateur et au dénominateur de la fraction. Son effet est donc nul.

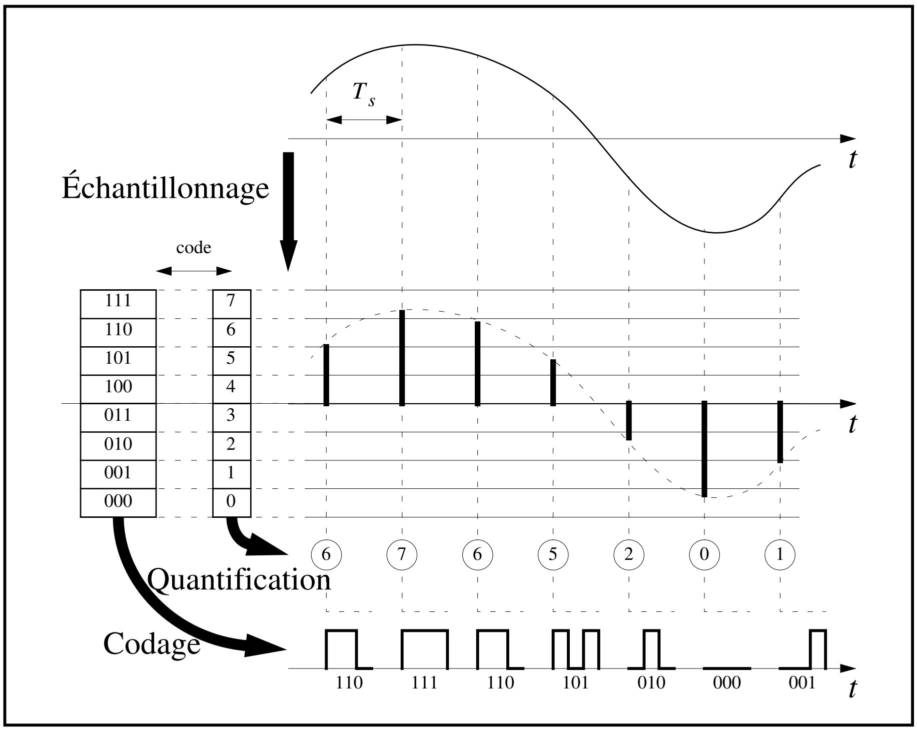
La fonction `linear2alaw(int pcm_val)` permet de convertir un échantillon PCM codé dans les 12 bits de poids forts d'un entier 16 bits en un caractère non signé (*unsigned char*) de 8 bits contenant le code de l'échantillon compressé selon la loi A.

La fonction `alaw2linear(unsigned char a_val)` permet de convertir un caractère non signé (*unsigned char*) de 8 bits en un échantillon décompressé selon la loi A dans les 12 bits de poids forts d'un entier 16 bits.

La fonction `search(int val, short *table, int size)` sert à identifier dans quel segment de la caractéristique de compression l'échantillon se trouve. Elle retourne la valeur du segment situé entre 0 et 7.

Je vais rappeler, ci-après et en quelques lignes, les principes de base de la numérisation d'un signal analogique. La figure 17 résume les différentes opérations de la numérisation proprement dite. Pour des raisons de clarté et de simplicité, le graphique propose une numérisation sur $2^3 = 8$

niveaux différents. Pour le cas d'un signal sonore dans les fréquences téléphoniques, le nombre de niveaux est de $2^{12} = 4096$. L'échantillonnage se fait 8000 fois par seconde. T_s représente la période d'échantillonnage.



T_s
t
Échantillonnage
code
111
110
101
100
011
010
001
000
7
6
5
4
3
2
1
0
t
6
7
6
5
2
0
1
Quantification
Codage
110
111
110
101
010
000
001
t

FIGURE 17. Étapes de la modulation PCM

Dans le domaine de la numérisation, il existe un théorème connu sous le nom de théorème de Shannon qui dit ceci:
Il est possible de reconstituer exactement un signal analogique quelconque pour autant que ses composantes fréquentielles soient limitées à la moitié de la fréquence de l'échantillonnage.

Cela signifie que, dans le cas de la transmission de sur une ligne téléphonique numérique, la fréquence maximale qu'il est possible de transmettre est de 4 kHz. Ce théorème est résumé par la formule suivante:

$$f_{max} \le \frac{f_s}{2}$$

où f_s est la fréquence de l'échantillonnage.

À la suite de cette numérisation, $12 \cdot 8000 = 96000$ bits seraient à transmettre en une seconde sur la ligne RNIS. Or le débit d'une telle ligne est limité à 64000 bits par secondes sur un canal B. C'est là qu'intervient la compression numérique. Le but est de créer artificiellement un pas de quantification plus fin pour les signaux de faible amplitude et plus large pour ceux de grande amplitude. En effet, l'oreille humaine est capable de percevoir des sons sur une dynamique très large. Par contre, sa résolution est relativement faible.

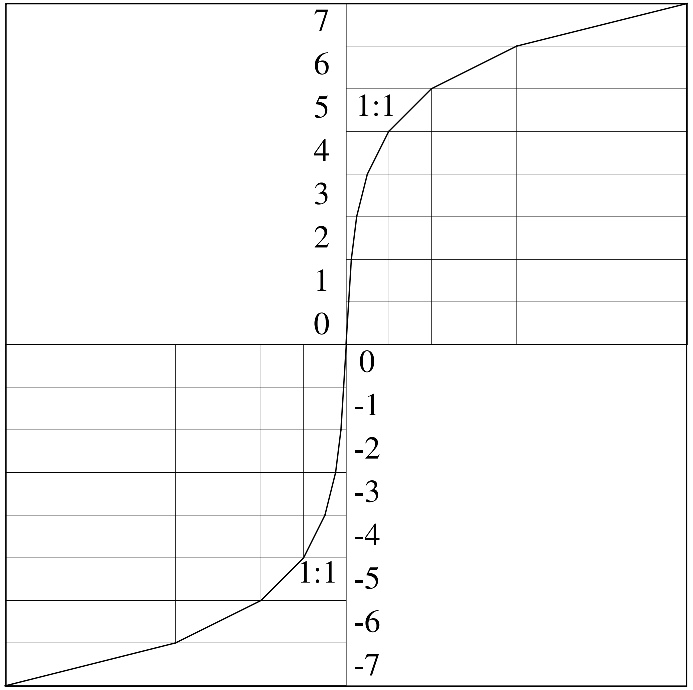
La loi A est une loi de compression qui tire en outre parti de la caractéristique de notre oreille. Théoriquement, elle est continue, mais pour les signaux numériques, elle est approchée par des segments de droite. Il y a 14 segments numéroté de -7 à +7, mais les segments positifs et négatifs portant l'indice 0 ont le même pas de quantification. C'est pourquoi l'on parle de loi de codage à 13 segments. La figure 18 présente ce principe. Tous les segments présentent 16 pas de quantification uniforme, sauf le segment central qui en contient 32.

Lors de la compression, on utilise les informations de signe, de numéro de segment et de valeur du signal pour composer un échantillon compressé sur 8 bits au lieu des 12 bits requis auparavant. C'est cet échantillon compressé qui est transmis sur la ligne à raison de 8000 échantillons par seconde. Dans ce cas, seuls $8000 \cdot 8 = 64000$ bits doivent être transmis. Il est alors possible d'utiliser le canal RNIS pour transmettre ces informations.

À la décompression, on extrait de l'échantillon les information de signe, de numéro de segment et l'information utile véhiculée par l'échantillon. On régénère alors l'échantillon original avec une perte de résolution. Les bits perdus lors de la compression sont simplement remplacés par des 0 lors de la décompression.

La compression permet de faire passer quasiment sans perte de qualité une séquence échantillonnée sur 12 bits au travers d'un canal de transmission qui ne supporte théoriquement qu'une numérisation sur 8 bits. La fonction de compression est une opération surjective, puisque à chaque échantillon codé sur 8 bits peut correspondre plusieurs échantillons codés sur 12 bits. De ce fait, la décompression d'un échantillon aboutit toujours à l'une des 256 valeurs autorisées parmi les 4096 possibles.

Caractéristique de compression à 13 segments selon la loi A



Code linéaire replié à 12 bits (Perte de résolution)	Code compressé à 8 bits	Code linéaire replié à 12 bits (Remplissage)
1 1 w x y z ▨ ▨ ▨ ▨ ▨ ▨	1 1 1 1 w x y z	1 1 w x y z 1 0 0 0 0 0
1 0 1 w x y z ▨ ▨ ▨ ▨ ▨	1 1 1 0 w x y z	1 0 1 w x y z 1 0 0 0 0
1 0 0 1 w x y z ▨ ▨ ▨ ▨	1 1 0 1 w x y z	1 0 0 1 w x y z 1 0 0 0
1 0 0 0 1 w x y z ▨ ▨ ▨	1 1 0 0 w x y z	1 0 0 0 1 w x y z 1 0 0
1 0 0 0 0 1 w x y z ▨ ▨	1 0 1 1 w x y z	1 0 0 0 0 1 w x y z 1 0
1 0 0 0 0 0 1 w x y z ▨	1 0 1 0 w x y z	1 0 0 0 0 0 1 w x y z 1
1 0 0 0 0 0 0 1 w x y z	1 0 0 1 w x y z	1 0 0 0 0 0 0 1 w x y z
1 0 0 0 0 0 0 0 w x y z	1 0 0 0 w x y z	1 0 0 0 0 0 0 0 w x y z
0 0 0 0 0 0 0 0 w x y z	0 0 0 0 w x y z	0 0 0 0 0 0 0 0 w x y z
0 0 0 0 0 0 0 1 w x y z	0 0 0 1 w x y z	0 0 0 0 0 0 0 1 w x y z
0 0 0 0 0 0 1 w x y z ▨	0 0 1 0 w x y z	0 0 0 0 0 0 1 w x y z 1
0 0 0 0 0 1 w x y z ▨ ▨	0 0 1 1 w x y z	0 0 0 0 0 1 w x y z 1 0
0 0 0 0 1 w x y z ▨ ▨ ▨	0 1 0 0 w x y z	0 0 0 0 1 w x y z 1 0 0
0 0 0 1 w x y z ▨ ▨ ▨ ▨	0 1 0 1 w x y z	0 0 0 1 w x y z 1 0 0 0
0 0 1 w x y z ▨ ▨ ▨ ▨ ▨	0 1 1 0 w x y z	0 0 1 w x y z 1 0 0 0 0
0 1 w x y z ▨ ▨ ▨ ▨ ▨ ▨	0 1 1 1 w x y z	0 1 w x y z 1 0 0 0 0 0

FIGURE 18. Compression et décompression selon la loi A

11.3.2 Fonction de recherche du maximum

Cette fonction implémente la recherche de la valeur maximale d'un tableau. Elle retourne dans une structure de résultat la valeur signée du maximum trouvé ainsi que son index. Elle sert à retrouver la valeur maximale de l'autocorrélation ou de la corrélation croisée.
La recherche se fait simplement en comparant la valeur absolue du premier élément du tableau avec la valeur absolue du second. La valeur maximale signée ainsi que son index sont stockés dans deux variables temporaires. Celle-ci est alors comparée à la valeur absolue du troisième élément du tableau et ainsi de suite.

Il est bien évident que ce type de recherche est relativement primaire et extrêmement sensible aux éventuelles erreurs de calcul. Une possibilité serait de prendre la moyenne des x derniers échantillons est de la comparer à l'échantillon courant. Si la valeur de ce dernier est grandement supérieur à la moyenne, on peut considérer qu'il s'agit d'une valeur erronée. Cela revient en fait à faire un filtrage passe-bas du signal.

L'appel de `smax(complex *buf, int length, maximum *max)` applique la recherche du maximum sur le tableau de complexes passé comme paramètre.

Le code de *maxsrch.c* est fourni sur la disquette d'annexes.

11.3.3 Fonction de retournement

Comme défini dans le chapitre "Concepts", le calcul de la corrélation est identique à celui de la convolution si ce n'est que l'une des deux séquences doit être retournée selon l'axe du temps. C'est pour cette raison que j'ai écrit la fonction `reverse(complex *buf, int length).` Son action revient simplement à retourner les échantillons contenus dans le tableau de complexes qui est fourni en paramètre.

Le code de *reverse.c* est disponible sur la disquette d'annexes.

11.4 Programmes auxiliaires

Pour me rendre la tâche plus facile, j'ai eu besoin de quelques petits programmes utilitaires. En voici une brève description.

11.4.1 Programme *chsize*

Ce programme sert simplement à copier un nombre donné de caractères (et par conséquent d'échantillons compressés) d'un fichier source dans un fichier de destination.

Il est utile pour raccourcir une séquence d'échantillon dont la longueur n'est pas une puissance de 2. Par contre, il ne permet pas de rallonger une séquence trop courte!
De plus, il supprime les 32 premiers octets du fichier source qui sont supposés contenir l'en-tête du fichier de son.

Le code du programme *chsize.c* est disponible sur la disquette d'annexes.

11.4.2 Programme *gensinc*

Cet outil génère une séquence d'échantillons linéaires selon la fonction sinc. La longueur de la séquence ainsi que le nom du fichier à créer sont donnés à l'appel du programme.

Le code du programme *gensinc.c* est disponible sur la disquette d'annexes.

11.5 La "makefile"

L'une des particularité du monde UNIX est que le compilateur C ou C++ est une commande en ligne. Par conséquent, il est possible, comme avec toutes les autres commandes, de créer des scripts d'exécution.

Pour la compilation d'un programme, le script consacré s'appelle "makefile". Il contient les options de compilation, les chemins pour la recherche des librairies, ainsi que les directives ou règles pour la compilation des modules.

Cette façon de faire donne toute la souplesse voulue au programmeur. La commande make permet d'exécuter ce script de compilation autant de fois que nécessaire. Lorsque une modification est apportée à l'un ou à plusieurs des codes sources impliqués, l'invocation de la commande *make* reconstruit tout ce qui est nécessaire à la prise en compte des modifications.

Dans le cadre de ce projet, vu l'utilisation de différents codes sources dans l'application, j'ai écrit une makefile afin de rendre la compilation des différents modules plus aisée.

La *makefile* est proposée sur la disquette d'annexes.

12. Tests

La fiabilité des différents parties du programme est primordiale. Il faut que chaque module ou fonctions donne un résultat correct pour obtenir quelque chose de valable à la fin du processus de traitement. C'est pour cette raison que j'ai entrepris de tester tous les modules séparément, puis réunis dans un seul et unique programme. Ce chapitre traite des essais mis en oeuvre pour assurer le bon fonctionnement des éléments du programme.

12.1 Code source

12.1.1 La FFT

Malgré le fait que le code de la FFT ait été écrit par une personne ayant fait son doctorat sur le sujet, j'ai dû tester la fiabilité de la classe effectuant les transformées de Fourier directes et inverses. J'ai pour cela généré quelques signaux simples dont la transformée de Fourier m'est connue. Puis je les ai transformés à l'aide de l'algorithme de FFT. J'ai ensuite comparé les résultats obtenus avec les résultats théoriques. Leurs concordances me permettent d'affirmer que le code de la FFT donne les résultats escomptés et par conséquent il est possible d'utiliser cette classe sans risque d'introduction d'erreur dans le processus de calcul des paramètres d'annulation de l'écho.

12.1.2 Les fonctions de conversion

Le code de base utilisé pour les fonctions de conversion de la loi A en linéaire et inversement est un code fourni par Sun Microsystems Inc. Bien qu'il soit largement utilisé, je dois m'assurer que son fonctionnement est correct. Pour cette raison, j'ai effectué des tests de compression et de décompression en utilisant d'une part l'algorithme de Sun et d'autre part en effectuant le calcul manuellement.
Cela m'a permis de détecter une petite anomalie dans la fonction `linear2alaw`. En effet, si on passe une valeur comprise entre -7 et 0 à la fonction, celle-ci retourne la valeur hexadécimale 0x5A alors que la valeur correcte qui devrait être retournée, toujours en hexadécimale, est 0x55. La petite correction que j'ai apporté au code permet de s'affranchir de ce problème.

Avec cette petite retouche, le code exécute parfaitement le travail qui lui est assigné. Il est utilisable sans réserves pour effectuer les opérations de compression et de décompression selon la loi A.

Le code *g711.c* corrigé est fourni sur la disquette d'annexes.

12.1.3 La fonction de recherche du maximum

Cette fonction ne comporte pas de réelle difficulté dans son implémentation. Elle est aussi relativement simple et par conséquent, ne présente pas de réelles possibilités d'erreurs. Chaque fois que j'ai utilisé cette fonction au cours de ce travail, elle a donné les résultats escomptés. Il est donc possible de l'utiliser sans souci.

12.1.4 La fonction de correction du signal

C'est la partie la plus délicate du code car c'est sur elle que repose tout le travail d'annulation de l'écho. Il faut en effet qu'elle corrige chaque échantillon de la bonne valeur sinon l'annulation n'est pas efficace.

La version actuelle de *cancel.c* est fournie sur la disquette d'annexes.

C'est la partie du code qui me pose malheureusement le plus de problèmes de mise au point. La mise en place d'une procédure de test n'est pas une chose aisée, mais voici celle que j'ai suivie dans le cadre de mon travail:

- test de la fiabilité de l'algorithme, tant au niveau du calcul du taux d'écho que du temps de décalage, à l'aide de signaux idéaux générés à l'aide du programme *gensinc*.
- répétition de la démarche précédente avec des signaux réel ayant subi les perturbations liées au canal de transmission.

Dans les deux prochains chapitres, je vais expliquer quels ont été les démarches entreprises lors des essais effectués tant avec les signaux idéaux qu'avec les signaux réels.

12.2 Correction de signaux idéaux

Comme décrit précédemment, la première étape dans le test de l'algorithme s'est déroulée à l'aide de signaux idéaux générés par le programme gensinc. Ces signaux ont les caractéristiques suivantes:

- durée utile de 512 échantillons
- durée totale de 4096 échantillons
- multiplication par un cosinus de fréquence première, en l'occurrence 809 Hz afin de décaler le spectre du signal dans la bande de fréquence téléphonique

Ce dernier point mérite quelques éclaircissements. La raison de la multiplication par une fréquence première est principalement d'éviter que le signal échantillonné contienne un nombre élevé de valeurs nulles. Si l'on utilisait par exemple une fréquence de 2 kHz pour le cosinus, un échantillon sur 2 serait nul. En effet, comme la fréquence de 2 kHz est un sous-multiple de la fréquence d'échantillonnage et qu'elle est en phase avec cette dernière, on obtient un échantillonnage semblable à ce qui est présenté à la figure 19. T_s représente la période d'échantillonnage.

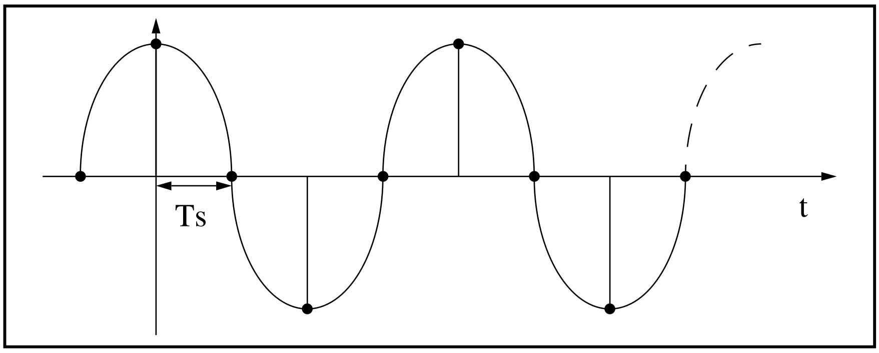


FIGURE 19. Numérisation d'un sous-multiple de la fréquence d'échantillonnage

Pour éviter ce phénomène, il est important de choisir une fréquence qui n'est pas multiple ou sous-multiple de la fréquence d'échantillonnage. De plus, il faut quand même que cette fréquence déplace le spectre du sinc dans la bonne bande de fréquence. C'est pourquoi j'ai choisi la fréquence 809 Hz. Elle présente l'intérêt de ne pas être multiple de la fréquence d'échantillonnage et de décaler le spectre dans la région de 800 Hz qui est la gamme de fréquences la plus présente dans le spectre de la voix humaine.

Le signal utilisé pour ces tests de signaux idéaux est présenté à la figure 20. Son spectre est quant à lui illustré par la figure 21.

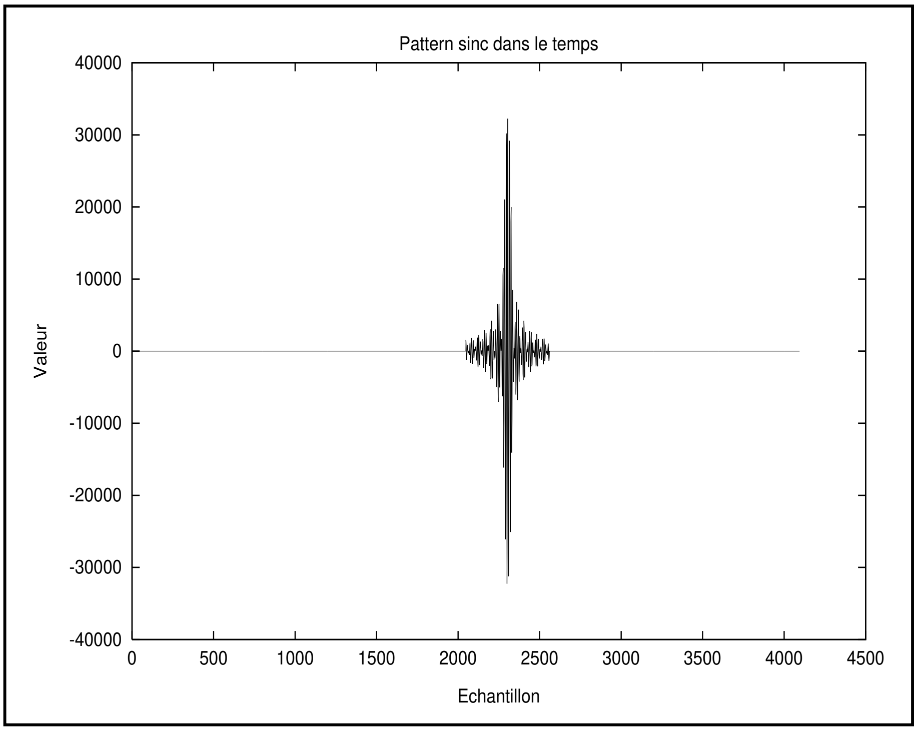


FIGURE 20. Représentation temporelle du pattern original

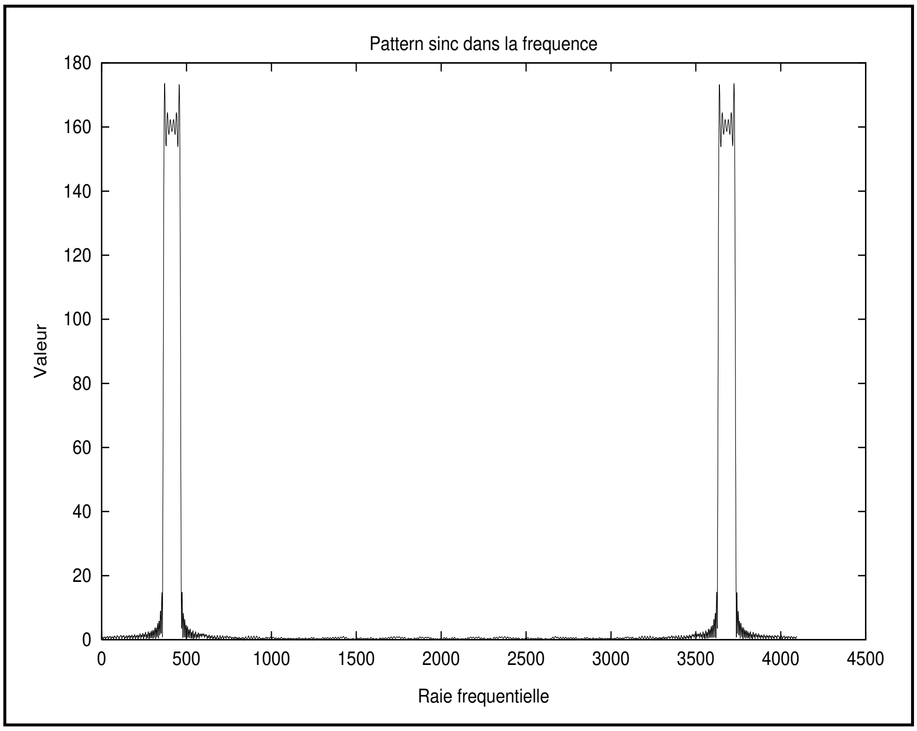


FIGURE 21. Représentation fréquentielle du pattern original

Pour pouvoir interpréter correctement les résultats retournés par le calcul de la FFT, je dois donner quelques explications relatives à la conversion des raies fréquentielles de la FFT discrète en valeurs fréquentielles réelles. Trois points importants sont à discuter:

- la raie fréquentielle n° 0 de la FFT représente la composante continue du signal (fréquence 0).
- le spectre discret est périodique de période inverse à la fréquence d'échantillonnage.
- le spectre est symétrique par rapport à la raie fréquentielle n° N/2.

La relation mathématique entre le numéro d'indice d'une raie fréquentielle est sa fréquence réelle est la suivante:

$$f_r = n_i \cdot \frac{f_s}{N}$$

avec f_r qui représente la fréquence réelle, f_s la fréquence d'échantillonnage du signal, N le nombre de points sur lequel la FFT se calcule, et n qui est l'indice de la raie fréquentielle en question.

Afin de tester les deux fonctionnalités de base du logiciel, j'ai généré de plus une copie du signal original, mais affaiblie d'un coefficient variable et décalée d'un nombre d'échantillons quelconque. Il est clair que ces deux coefficients me sont connu. Le but du test est de voir si le programme est capable de les retrouver uniquement par le calcul.

Après plusieurs essais avec différents coefficients d'écho et valeurs de décalage, je peux affirmer que dans tous les cas, l'algorithme à été capable de retrouver les paramètres utilisés. C'est déjà là un bon point en vue de l'application de cette méthode à des signaux réels.
De plus, la correction du signal donne aussi de bons résultats sur ces signaux idéaux. Le signal corrigé est totalement épuré du signal original. Dans ce cas, le fonctionnement de la méthode est optimal.

Mais le but de ce travail est de traiter des signaux réels ayant transité par un canal de transmission téléphonique et là, les choses ont tendance à se compliquer quelque peu. Dans le prochain chapitre, je vais traiter de la correction des signaux réels et mettre en lumière les difficultés qui se présentent lorsque l'on passe du cas idéal au cas pratique.

12.3 Correction de signaux réels

Pour le test de l'algorithme en présence de signaux réels, j'ai utilisé le même pattern que pour le cas idéal (voir figure 20). Par contre, pour ce qui est du signal d'écho, il est dans ce cas enregistré directement sur la ligne téléphonique. C'est à dire que le pattern original est diffusé sur le canal de transmission, et simultanément, à l'aide de l'application Mail-Vox, son écho est enregistré. C'est ce dernier fichier qui va servir à effectuer les calculs de recherche du taux d'écho et du décalage. La figure 22 présente le signal temporel de l'écho du pattern émis qui a été enregistré par l'application MailVox.

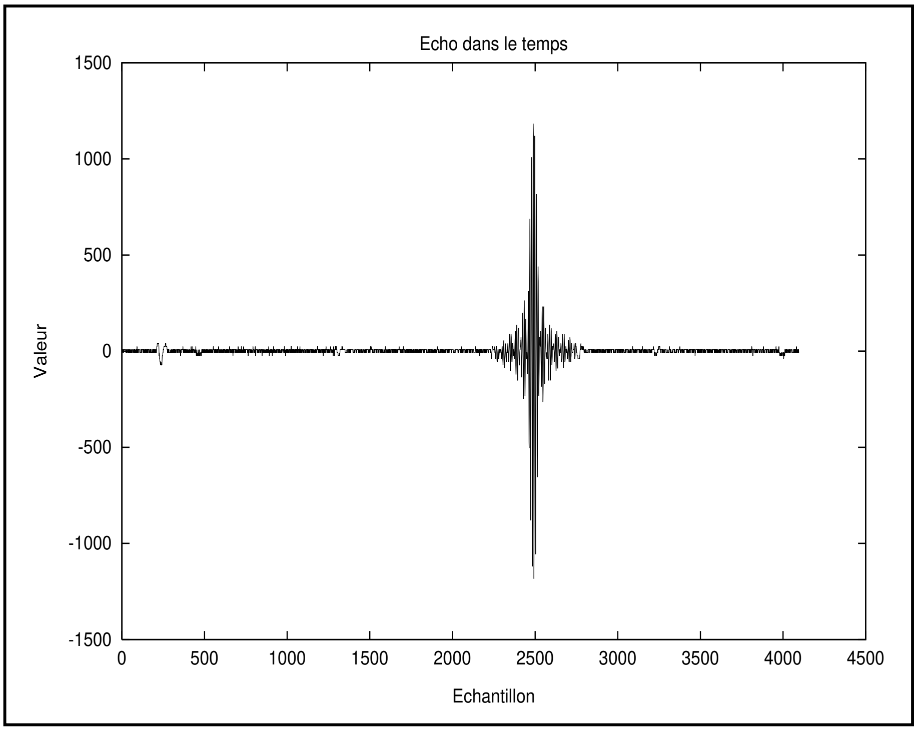


FIGURE 22. Représentation temporelle du signal enregistré

La figure 23 quant à elle donne la représentation fréquentielle de ce même signal.

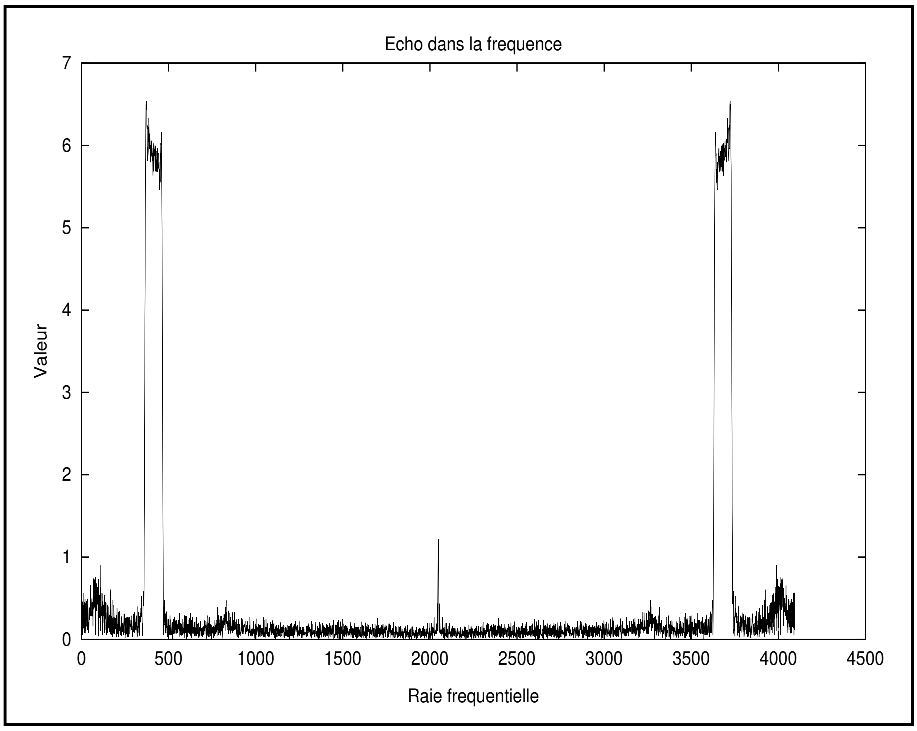


FIGURE 23. Représentation fréquentielle du signal enregistré

12.3.1 L'impulsion de taxe

Le premier problème que j'ai rencontré est lié à la présence des impulsions de taxes. En effet, une impulsion de taxation du réseau téléphonique est normalisée à 12 kHz. Théoriquement, cette fréquence ne devrait pas être numérisée dans le signal puisqu'elle est au delà de la fréquence maximale définie par le théorème de Shannon. Mais pratiquement, on retrouve une fréquence de 4 kHz dans le signal numérisé. Cette fréquence est clairement visible au milieu du spectre du signal enregistré (voir la figure 23). Pour quelle raison et pourquoi 4 kHz?
L'explication de la fréquence est simple. La fréquence d'échantillonnage est de 8 kHz. La fréquence de l'impulsion de taxe est de 12 kHz. Or la numérisation d'un signal de trop haute fréquence revient à numériser la différence des deux fréquences modulo la fréquence d'échantillonnage, dans ce cas 4 kHz.
Par contre, la raison pour laquelle cette fréquence apparaît n'est pas évidente. Il faut noter que dans le cadre de ce projet, la partie analogique de la liaison est en fait un port analogique du NT (*Network Termination*) d'une ligne RNIS. Par conséquent, l'information de taxation circule entre le central PTT et le NT de la ligne Swissnet sous forme numérique. C'est le NT lui-même qui régénère le 12 kHz de l'impulsion de taxe sur le terminal analogique lorsqu'il reçoit un paquet de taxation.

Un filtre *antialiasing* servant à couper les fréquences supérieures à la moitié de la fréquence d'échantillonnage est inséré avant le convertisseur analogique-numérique d'un port analogique du NT. Donc cette fréquence ne devrait pas apparaître. Pour m'en convaincre, j'ai procédé à l'expérience suivante. J'ai produit à l'aide d'un générateur de signaux une onde sinusoïdale de fréquence 12 kHz que j'ai diffusé à l'aide d'un haut-parleur en direction du microphone de la station téléphonique et, simultanément, j'ai enregistré la réponse dans un fichier sur la station Sun (figure 24). Le but de cette expérience est de vérifier le bon fonctionnement du filtre *antialiasing* du NT.

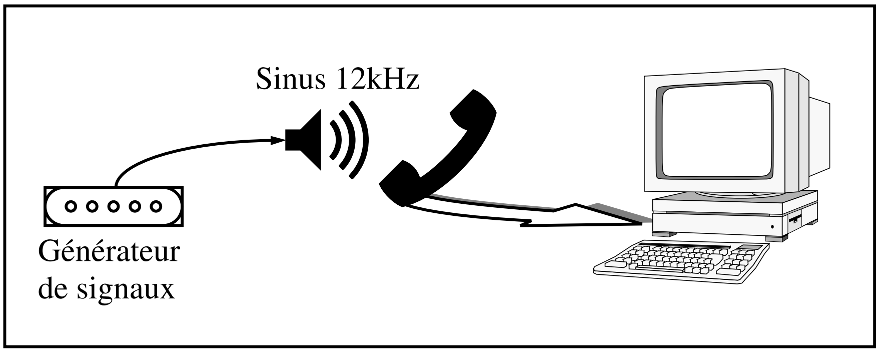


FIGURE 24. Principe de test du filtre *antialiasing* du NT

Le résultat est conforme à mon attente, à savoir qu'aucune fréquence n'est audible dans le signal reçu par la station Sun. Le filtre *antialiasing* du NT fonctionne correctement. Le problème se situe donc en amont de

ce filtre.
L'hypothèse la plus vraisemblable est que le signal filtré est "pollué" par le 12 kHz entre la sortie du filtre et le convertisseur A/D. Malheureusement, il n'est pas possible de supprimer les impulsions de taxes dans le type de NT installé à l'école afin de vérifier cette hypothèse.

12.3.2 Sensibilité au décalage

Le point suivant qui m'a posé des problèmes est la sensibilité de l'algorithme aux erreurs de calcul du décalage. Si le bruit présent sur la ligne modifie la valeur des échantillons de manière significative, il se peut que le calcul du décalage soit erroné. Or un décalage d'un seul échantillon peut annihiler totalement l'effet de la correction. En effet, la norme de l'échantillon peut fortement varier d'un échantillon à l'autre (figure 25). Donc si un décalage intervient dans le calcul des paramètres, il y a de fortes chances pour que l'on tombe sur un échantillon dont la valeur est éloignée de la vraie valeur et ainsi tout le calcul du taux d'écho est faussé.

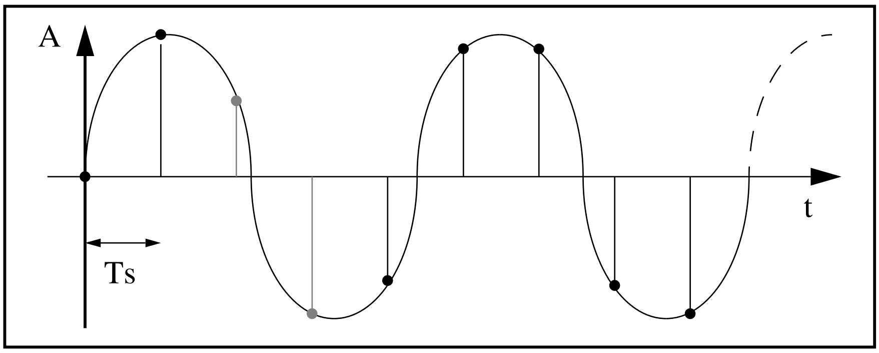


FIGURE 25. Sensibilité au décalage

Pour contrer ce phénomène, il faudrait que le signal du pattern utilisé présente plusieurs valeurs proches autour du maximum de la fonction pour assurer que, même en cas de faible décalage, la valeur calculée du taux d'écho soit proche de la vraie valeur. Or cela va à l'encontre des spécifications émises quant au choix du signal. En effet, celui-ci doit présenter un maximum franc dans sa fonction de corrélation. Par contre, il est possible de diminuer au maximum la fréquence de l'impulsion sinc afin d'obtenir un nombre plus important d'échantillons dans la région du maximum de cette fonction
Une autre solution serait d'interpoler le signal dans la région du maximum, puis de le sur-échantillonner pour obtenir plus de points et ainsi une meilleure résolution.

13. Résultats et propositions

Je vais récapituler dans ce chapitre les points forts et les points faibles des résultats obtenus lors de la réalisation de ce projet de diplôme. Je veux surtout traiter avec attention les difficultés qui sont apparues durant le déroulement du travail, afin de permettre la mise à profit ces constatations dans le futur. De plus, quelques propositions concernant la suite possible de ce projet seront abordées.

Je me suis heurté à quelques difficultés qui ont perturbé le déroulement du travail. Ces problèmes m'ont empêché d'atteindre l'objectif final du projet, à savoir l'annulation d'écho sur des signaux réels ayant transité par le réseau téléphonique public. Ces difficultés sont principalement dues à la sensibilité de l'algorithme aux erreurs de décalage du maximum des fonctions de corrélation. Cela provient du fait que les échantillons proches du maximum de la fonction de corrélation ont une variation importante de leur valeur relative pour la fonction sinc.
Je n'ai pas trouvé la fréquence de la fonction sinc qui me donne le meilleur compromis entre le nombre d'échantillons proches du maximum et la meilleure couverture spectrale du signal.

Il va sans dire que le fait de diminuer la fréquence du sinc ne permet plus de couvrir une large bande spectrale. Il est donc difficile de conjuguer cette méthode d'annulation d'écho avec la recherche de la fonction de transfert du canal de transmission. Cette dernière requiert en effet un spectre uniforme dans la bande des fréquences téléphoniques (300-3400 Hz).

De plus, le modèle décrit dans les hypothèses de travail est certainement trop simplifié, notamment en ce qui concerne les distorsions fréquentielles appliquées au signal qui parcourt le canal de transmission. Il serait utile, à l'avenir, de prendre en compte une modélisation plus fine du canal de transmission et éventuellement d'appliquer une égalisation au signal avant de commencer tout traitement d'annulation de l'écho.

Heureusement, j'ai aussi obtenu quelques résultats positifs au cours de ce travail. Tout d'abord, la mise en place des outils logiciels permettant d'analyser les signaux émis et reçus par le serveur est pleinement remplie. Cela comprend les travaux suivants:

- modification de l'application MailVox afin de lui adjoindre la capacité de full-duplex. Ceci autorise maintenant la diffusion et l'enregistrement simultanés de messages sonores.
- création du logiciel permettant d'effectuer les transformées de Fourier des signaux vocaux en vue de calculer les fonction d'autocorrélation et de corrélation croisée sur ces signaux. Ajouté à cela, il y a aussi l'écriture des différents modules auxiliaires permettant le retournement d'une

  séquence sonore, la compression et la décompression selon la loi A ainsi que la recherche du maximum des fonctions de corrélation.
- l'écriture de programme auxiliaires servant à générer des signaux idéaux importants pour l'étude des phénomènes d'écho et du comportement de l'algorithme d'annulation.

Ensuite, les tests du logiciel à l'aide des signaux idéaux ont démontré que la méthode utilisée en vue de l'annulation de l'écho est envisageable. Le fait que les résultats obtenus avec ces signaux idéaux sont positifs est un encouragement à poursuivre dans cette voie.

Le programme d'extraction des paramètres remplis la contrainte de temps qui prévoit que le calcul doit s'effectuer en moins d'une seconde. Dans la totalité des essais que j'ai mis en oeuvre, le temps de calcul était même inférieur à une demi seconde pour des opérations portant sur 4096 points, soit environ 500ms de signal. Par expérience, cette durée de signal est suffisante pour permettre l'extraction correcte des paramètres de correction.

Finalement, l'écriture du logiciel sous la forme de petits modules indépendants mais complémentaires permet d'obtenir toute la souplesse voulue en vue des modification ultérieures du code source de l'application.

14. Conclusion

Le monde des serveurs vocaux est un domaine passionnant que j'ai eu la chance de côtoyer dans le contexte de ce travail de diplôme. J'ai eu beaucoup de plaisir à explorer ce domaine spécifique du traitement de signal et de l'informatique.

Ce travail a en outre soulevé un nombre important de points qui restent à éclaircir. Malgré le manque de résultats probants au niveau de l'annulation de l'écho sur des signaux réels, je garde l'espoir que la voie suivie débouchera sur les résultats escomptés. Il faut persévérer et approfondir les points qui m'ont posé des problèmes.

C'est pour cette raison que les outils nécessaires à l'étude de la problématique de l'écho ont été créés. Ils se déclinent sous trois ensembles.
Le premier qui est la modification de l'application MailVox pour permettre la diffusion et l'enregistrement simultanés de messages vocaux. Il devient ainsi possible d'enregistrer l'écho d'un signal diffusé.
Le second qui contient toute la création de l'algorithme de base pour effectuer les transformées de Fourier et le calcul des opérations de corrélation. De plus un shell-script permet de visualiser sous forme graphique les différents opérations effectuées par l'algorithme.
Enfin le troisième qui couvre les différents programmes auxiliaires permettant de générer aisément des signaux idéaux.

Je souhaite à celui ou celle qui poursuivra mes travaux de trouver dans ce rapport toutes les informations nécessaires à la bonne continuation de ce projet et surtout d'avoir autant de plaisir que moi à explorer ce domaine de la science.

Martigny, le 6 décembre 1996

Florian Salamin

15. Remerciements

Malgré son apparence individuelle, ce travail de diplôme est en fait le fruit d'une étroite collaboration entre le personnel de l'institut IDIAP, les professeurs de l'École d'Ingénieurs du Valais et moi-même.
C'est pour cette raison que je tiens à remercier tout particulièrement les personnes suivantes:

- M. Olivier Bornet, ingénieur de recherche à l'IDIAP pour son soutien, son aide et sa disponibilité lors de ce travail. Ses connaissances du monde UNIX et de Solaris m'ont été fort utiles,
- M. François Corthay, professeur à l'École d'Ingénieurs du Valais, pour ses conseils concernant le déroulement du travail, ainsi que ses connaissances approfondies du logiciel MatLab,
- M. Dominique Gabioud, professeur à l'École d'Ingénieurs du Valais, pour ses informations pertinentes sur les systèmes de télécommunications,

ainsi que le personnel de l'École d'Ingénieurs du Valais.

En plus de ces personnes qui ont participé au bon déroulement du projet au niveau technique, il y a aussi tout ceux qui ont contribué de manière directe ou indirecte à la réalisation de ce projet. C'est notamment la cas de:

- M. Jean-Claude Salamin,
- M. Gilles Burnier,
- M. Jean-Christian de Rivaz,
- M. Willy Rossier,
- Mlle Anouchka Primmaz,
- et ma famille.

Sans l'aide de toutes ces personnes, rien n'aurait été possible. Qu'elles en soient remerciées.

16. Bibliographie

16.1 Ouvrages

[1] *Systèmes de télécommunications*, p.216-226, Pierre-Gérard Fontolliet, Presses polytechniques et universitaires romandes, 1996.

[2] *Techniques modernes de traitement numérique des signaux*, p217-278, sous la direction de Murat Kunt, Presses polytechniques et universitaires romandes, 1991.

[3] Cours de télécommunications, Dominique Gabioud, École d'Ingénieurs du Valais, 1996.

[4] *The UNIX programming environment*, Brian W. Kernighan et Rob Pike, Prentice-Hall software series, 1984.

[5] *The C programming language*, Brian W. Kernighan et Dennis M. Ritchie, Prentice-Hall software series, 1988.

[6] *Le langage C++*, Bjarne Stroustrup, Addisson-Wesley, seconde édition 1992.

[7] Cours de programmation C++, Marylène Micheloud et Medard Rieder, École d'Ingénieurs du Valais, 1996.

[8] *Essential System Administration*, Aeleen Frisch, O'Reilly & Associates Inc., 1991.

[9] UNIX system V, manuel de référence du gestionnaire système, traduit de l'anglais par AT&T, Prentice-Hall, 1987.

[10] UNIX system V, manuel de référence de l'utilisateur, traduit de l'anglais par AT&T, Prentice-Hall, 1987.

[11] *Nouveau mémento UNIX system V*, Thierry Lafue et Jean-Baptiste Piacentino, Sybex, 1989.

[12] *XTL architecture guide*, Sun Microsystems Inc., 1994.

[13] *XTL application programmer's guide*, Sun Microsystems Inc., 1994.

[14] VIA, le magazine du rail, numéro 4/96 page 24, CFF, 1996.

16.2 Ressources Internet

[15] Forum de discussion:

comp.speech.

[16] Pages d'algorithmes de FTT dans différents langages dont le C++:

http://www.spektracom.de/~arndt/fxt/fxtpage.html

http://www.archelon.com/fft.html

[17] Différentes pages d'algorithmes mathématiques:

http://www.netlib.org/

http://www.cs.sunysb.edu/~algorith/files/fourier-transform.shtml

http://www.roe.ac.uk/computing/starlink/docs/sun194.htx/sun194.html#stardoccontents

[18] Quelques liens concernant la programmation:

http://www.utu.fi:80/~sisasa/oasis/

http://www.voicenet.com/tech/comp/prog/

[19] Codage numérique:

http://www-mobile.ecs.soton.ac.uk/speech_codecs/standards/adpcm.html

http://www.itl.atr.co.jp/comp.speech/Section3/speechlinks.html

http://wwwdsp.ucd.ie/speech/speech_code.html

Table des figures

Principe de fonctionnement 1
Planning 4
Provenance de l'écho 7
Modification full-duplex 9
L'environnement SunXTL 10
Atténuation et décalage dans le temps 13
Calcul de la corrélation 17
Calcul du taux d'écho 19
Extrait de voix 23
Fonction sinus cardinal 24
Comparaison du nombre d'opérations pour la DFT et la FFT 27
Résultat de la simulation dans le domaine du temps 28
Résultat de la simulation dans le domaine des fréquences 29
Comparaison des temps de calcul 31
Déroulement temporel du processus de correction 32
Découpage du logiciel 33
Étapes de la modulation PCM 37
Compression et décompression selon la loi A 39
Numérisation d'un sous-multiple de la fréquence d'échantillonnage 44
Représentation temporelle du pattern original 44
Représentation fréquentielle du pattern original 45
Représentation temporelle du signal enregistré 47
Représentation fréquentielle du signal enregistré 47
Principe de test du filtre antialiasing du NT 48
Sensibilité au décalage 49