II- La reconnaissance vocale.

I- Historique de la reconnaissance vocale.

La reconnaissance vocale est un outil récent. Vers 1950 apparut le premier système de reconnaissance de chiffres, appareil entièrement câblé et imparfait. Vers 1960, l'introduction des méthodes numériques et l'utilisation des ordinateurs changent la dimension des recherches.

Néanmoins les résultats demeurent modestes car la difficulté du problème avait été largement sous-estimée, en particulier en en ce qui concerne la parole continue. Vers 1970, la nécessité de faire appel à des contraintes linguistiques dans le décodage automatique de la parole avait été jusque-là considérée comme un problème d'ingénierie. La fin de la décennie 70 voit se terminer la première génération des systèmes commercialisés de reconnaissance de mots. Les générations suivantes, mettant à profit les possibilités sans cesse croissantes de la micro-informatique, possèderont des performances supérieures (système multi locuteurs, parole continue).

On peut résumer en quelques dates les grandes étapes de la reconnaissance de la parole (cf. techniques de l'ingénieur, vol.H1 940, p.3):

- 1952: Reconnaissance des 10 chiffres, pour un mono locuteur, par un dispositif électronique câblé.
- 1960: Utilisation des méthodes numériques.
- 1965: Reconnaissance de phonèmes en parole continue.
- 1968: Reconnaissance de mots isolés par des systèmes implantés sur gros ordinateurs (jusqu'à 500 mots).
- 1969: Utilisation d'informations linguistiques.
- 1971: Lancement du projet ARPA aux USA (15 millions de dollars) pour tester la faisabilité de la compréhension automatique de la parole continue avec des contraintes raisonnables.
- 1972: Premier appareil commercialisé de reconnaissance de mots.
- 1976: Fin du projet ARPA; les systèmes opérationnels sont HARPY, HEARSAY I et II et HWIM.

II- Le son.

Bien comprendre les processus tels que le codage du son dans l'ordinateur ou sa synthèse, c'est d'abord bien comprendre le son lui-même. Nous commencerons donc par définir le son.

a) *Qu'est ce que le son ?*

Le son est une vibration de l'air. A l'origine de tout son, il y a mouvement (par exemple une corde qui vibre, une membrane de haut-parleur...). Il s'agit de phénomènes oscillatoires créés par une source sonore qui met en mouvement les molécules de l'air. Avant d'arriver jusqu'à notre oreille, ce mouvement se transmet entre les molécules à une vitesse de 331 m/s à travers l'air à une température de 20°C : c'est ce que l'on appelle la propagation.

Un son est d'abord défini par son volume sonore et sa hauteur tonale. Le volume dépend de la pression acoustique créée par la source sonore (le nombre de particules d'air déplacées). Plus elle est importante et plus le volume est élevé. La hauteur tonale est définie par les vibrations de l'objet créant le son. Plus la fréquence est élevée, plus la longueur d'onde est petite et plus le son perçu est aigu. En doublant la fréquence d'une note, on obtient la même à l'octave supérieure. Et donc, en divisant la fréquence par deux, on passe à l'octave inférieure. Ce n'est qu'au-delà de 20 vibrations par seconde que l'oreille perçoit un son. Les infrasons, de fréquence inférieure à cette limite de 20 Hz sont inaudibles, de même que les ultrasons, de fréquence supérieure à 20 000 Hz, soit un peu plus de 10 octaves. Chacun peut constater que le niveau sonore diminue à mesure que l'on s'éloigne de la source. Cette diminution est la même chaque fois que la distance est doublée. Cependant, les hautes fréquences ne se propagent pas aussi loin que les sons graves. Il faut plus d'énergie pour restituer les basses que les aigus.

b) Quelles sont les caractéristiques d'un son ?

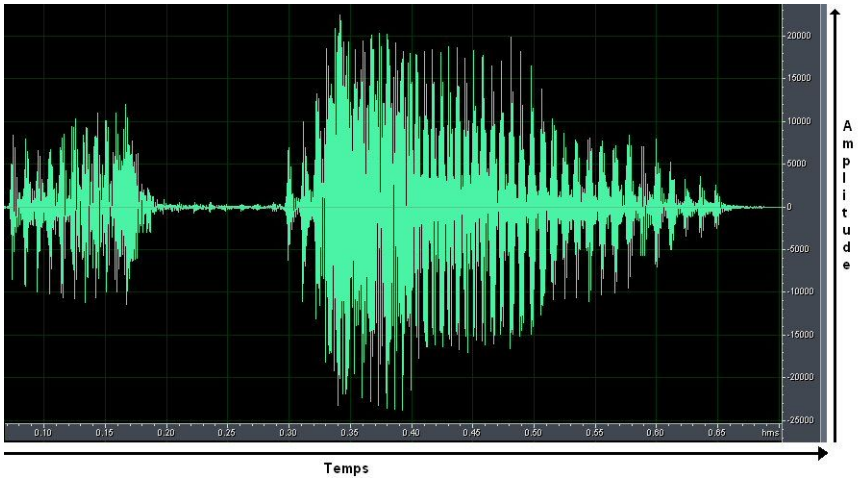
Le son est défini par trois paramètres :

- L'amplitude d'un son qui correspond à la variation de pression maximale de l'air engendrée par les oscillations, c'est-à-dire volume sonore

- La dynamique qui permet la mesure de l'écart entre le volume maximal d'un son et le bruit de fond. La dynamique ce mesure en décibels (dB). En fait, elle est le rapport entre le niveau maximal, à la limite de la distorsion, et le niveau minimal acceptable, à la limite du niveau de bruit de fond. Par exemple, pour un compact disque on parle de 90 dB de dynamique utile et en télévision de 25 dB en moyenne.

- Le timbre qui est un paramètre beaucoup plus subjectif : il s'agit de ce qui différencie deux sons de même hauteur et de même amplitude. C'est une notion qualitative, qui fera dire, par exemple, qu'un son est brillant ou profond…

On peut visualiser ces paramètres de plusieurs façons, la plus simple étant sa représentation bidimensionnelle.
Une représentation courante est l'amplitude de l'onde en fonction du temps :



c) Comment est stocké le son sur l'ordinateur ?

Il faut d'abords différencier les deux types de sons: le son analogique et le son numérique. Le son analogique est représenté sous la forme de signaux électriques d'intensité variable. Ces signaux sont issus d'un micro qui transforme le son acoustique d'une voix ou la vibration des cordes d'une guitare en impulsions électriques. Ces signaux sont enregistrables tels quels sur une bande magnétique (K7 audio par exemple) et peuvent être ensuite amplifiés, puis retransformés en son acoustique par des haut-parleurs. Le son analogique n'est pas manipulable tel quel par un ordinateur, qui ne connaît que les 0 et les 1.

Le son numérique est représenté par une suite binaire de 0 et de 1. L'exemple le plus évident de son numérique est le CD audio. Le processus de passage du son analogique en son numérique est appelé "échantillonnage". Celui-ci consiste à mesurer la tension (en Volt) du signal analogique à intervalles réguliers. La valeur obtenue est enfin codée en binaire (suite de 0 et de 1).Le composant qui réalise cette tache est appelée convertisseur A/N. Évidemment, ce processus de mesure et de conversion binaire doit être très rapide. C'est là qu'intervient la fréquence du son à numériser. Par exemple, pour une voix dont la fréquence est de 8000 Hz (Hertz), le signal électrique issu du micro aura aussi une fréquence de 8000 Hz. Pour transformer ce signal en numérique et à qualité équivalente, le mathématicien Shannon a démontré qu'il fallait que le prélèvement de mesures soit fait à une fréquence au moins 2 fois plus rapide que la fréquence originale, soit pour l'exemple de la voix, 16000 fois par seconde (16000 Hz ou 16 kHz). Un autre paramètre très important de l'échantillonnage est la précision avec laquelle la tension du signal électrique sera lue et codée. Le codage peut, en effet se faire sur 2 n bits. Une précision de 8 bits donnera une tension codée parmi 256 valeurs possibles, alors que 16 bits donneront 65 536 valeurs. Les CD sont ainsi échantillonnés à 44,1 kHz sur 16 bits.

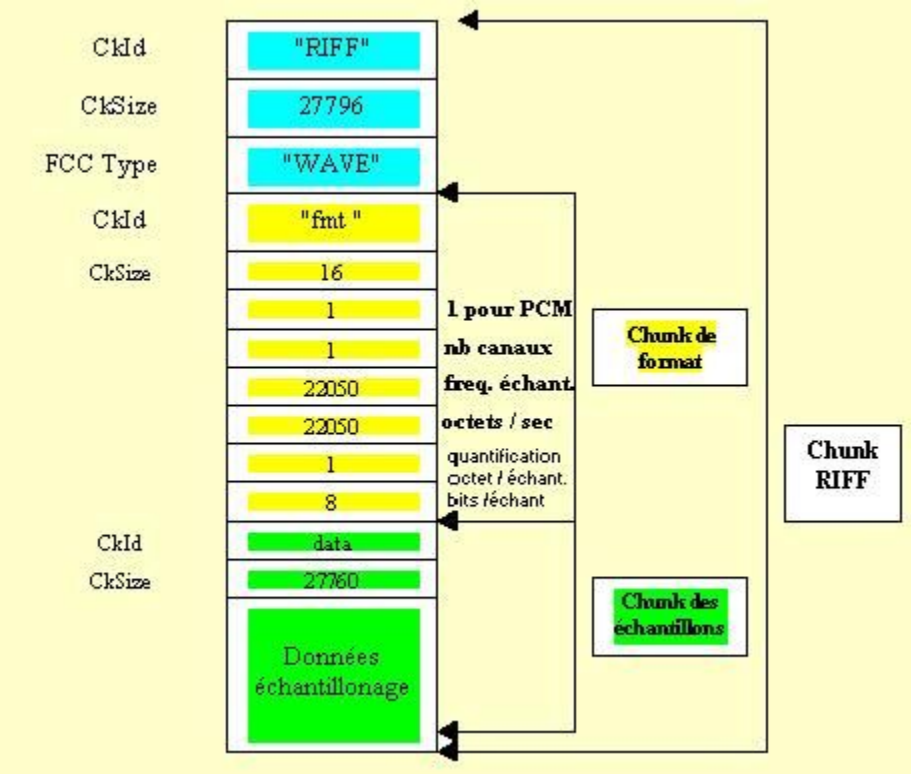
La conversion inverse de numérique vers analogique, se fait par le processus inverse. La tension du signal électrique est recréée à partir des valeurs codées, lues à la même vitesse qu'elles avaient été enregistrées. L'avantage évident de ce type de son, c'est qu'étant codé sous la forme de 0 et de 1, il est directement manipulable par un ordinateur et son stockage ne pose aucun problème sur un disque dur. En revanche, le nombre de valeurs enregistrées étant énorme (44 100 valeurs/s), ce type de son occupe beaucoup de place dans la mémoire ainsi que sur le disque dur de l'ordinateur. A titre d'exemple, un CD audio de 74 minutes représente 650 Mo (Mégaoctets) et

le débit d'un lecteur de CD est de 150 Ko/s (Kilo-octets/s). Une seule seconde de son stéréo échantillonné à 44,1 kHz et en 16 bits prend 172 Ko.

III- Qu'est-ce qu'un fichier audio numérique.

Notre reconnaissance vocale se base sur la comparaison de fichiers audio ainsi nous devons tout d'abord maîtriser le format d'enregistrement utilisé avant d'effectuer des opérations de transformation du signal.

On distingue deux types de format, les formats compressés et les formats non compressés, pour notre part nous avons utilisé un format non compressé qui est le format WAV (format largement utilisé depuis l'apparition de Windows). La norme de qualité du format WAV est 44.100 KHz à 16 Bits/s qui correspond à l'échantillonnage des CDs Audio. Sa structure est très simple, pour cela nous avons modélisé la structure du fichier 'tada.wav' :

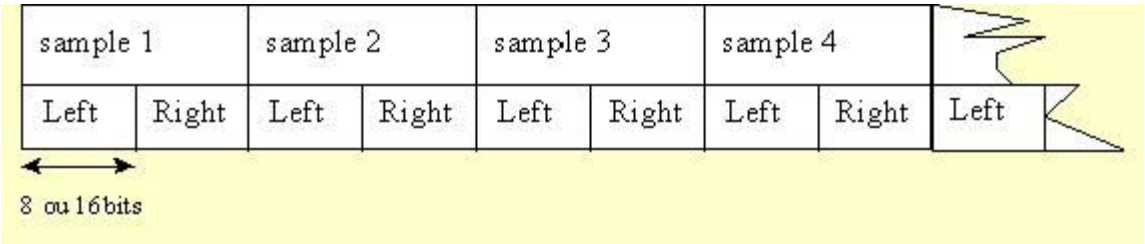


Le fichier est constitué de blocs hiérarchisés appelés "chunk". Le bloc "Riff " ("Ressource Interchange File Format", qui est le type des fichiers WAV) englobe l'ensemble du fichier, il permet de l'identifier comme étant un fichier WAV. Le bloc Format identifie les différents paramètres du format: fréquence de l'échantillonnage, nombre de bits etc. Le bloc data contient les données échantillonnées. Tous les champs sont facilement compréhensibles avec les notions vues précédemment, excepté la valeur du champ correspondant au format Microsoft qui est à 1 pour le format PCM, Pulse Code Modulation, format des données audio non compressés.

· Les données du son, les échantillons, sont alignés les uns après les autres dans l'axe des temps (dans l'ordre où ils arrivent dans le temps). En stéréo, les canaux sont multiplexés (entrelacés).

Ce multiplexage offre deux avantages : lecture/écriture des deux canaux en une seule opération disque (évite surtout les déplacements de têtes du disque) et l'augmentation aisée du nombre de canaux tout en restant dans la norme du format (par exemple un format quadriphonique).

Exemple de stockage des samples pour un son stéréo 8 bits ou 16 bits stéréo:



· Codage des échantillons :

Le codage est différent en 8 bits et en 16 bits. En 16bits, le mot est conforme à un entier signé 16bits (-32768, +32767) centré sur zéro. En 8 bits, l'octet est en fait un BYTE (0 - 255) centré sur 128. Pour transformer ce BYTE en entier signé par exemple, il faudra lui retrancher la valeur 128. Nous aurons alors un entier (-128 à +127) centré sur zéro correspondant ainsi au signal réel (devant être traité).

IV- La reconnaissance vocale.

La reconnaissance automatique de la parole est un domaine de la science ayant toujours eu un grand attrait auprès des chercheurs comme auprès du grand public. En effet, qui n'a jamais rêvé de pouvoir parler avec une machine ou, du moins, piloter un appareil ou un ordinateur par la voix. Ne plus avoir à se lever pour allumer ou éteindre tel ou tel appareil électrique, ne plus avoir à taper pendant des heures sur un clavier pour rédiger un rapport (par exemple). L'homme étant par nature paresseux, une telle technologie a toujours suscité chez lui une part d'envie et d'intérêt, ce que peu d'autres technologies ont réussi à faire.

Le secteur de la reconnaissance automatique de la parole est en pleine croissance et nous verrons dans ce document que la technologie actuelle est très aboutie, pouvant commencer à répondre aux attentes de l'homme. Bien que des progrès soient encore à faire sur les systèmes complexes de reconnaissance, il est à noter que la reconnaissance de petits vocabulaires est quasiment parfaite, ce qui suffit largement pour des outils de traitements vocaux du quotidien. Sans compter le coût de ces systèmes qui a considérablement chuté ces dernières années mais aussi le gain qu'ils peuvent apporter à un particulier et surtout à une entreprise.

a) Définition.

La reconnaissance automatique de la parole est l'un des deux domaines du traitement automatique de la parole, l'autre étant la synthèse vocale. La reconnaissance automatique de la parole permet à la machine de comprendre et de traiter des informations fournies oralement par un utilisateur humain. Elle consiste à employer des techniques d'appariement afin de comparer une onde sonore à un ensemble d'échantillons, composés généralement de mots mais aussi, plus récemment, de phonèmes (unité sonore minimale). En revanche, le système de synthèse de la parole permet de reproduire d'une manière sonore un texte qui lui est soumis, comme un humain le ferait. Ces deux domaines et notamment la reconnaissance vocale, font appel aux connaissances de plusieurs sciences : l'anatomie (les fonctions de l'appareil phonatoire et de l'oreille), les signaux émis par la parole, la phonétique, le traitement du signal, la linguistique, l'informatique, l'intelligence artificielle et les statistiques.

Il faut bien distinguer ces deux mondes : un système de synthèse vocale peut très bien fonctionner sans qu'un module de reconnaissance n'y soit rattaché. Evidemment le contraire est également tout à fait possible. Par contre, dans certains domaines bien précis, l'un ne va pas sans l'autre. Le traitement automatique de la parole ouvre des perspectives nouvelles, compte tenu de la différence considérable existant entre la commande manuelle et vocale. L'utilisation du langage naturel dans le dialogue personne/machine met la technologie à la portée de tous et entraîne sa vulgarisation, en réduisant les contraintes de l'usage des claviers, souris et codes de commandes à maîtriser. En simplifiant le protocole de dialogue personne/machine, le traitement automatique de la parole vise donc aussi un gain de productivité puisque c'est la machine qui s'adapte à l'homme pour communiquer, et non l'inverse. De plus, il rend possible l'utilisation simultanée des yeux ou des mains à une autre tâche. Il permet d'humaniser les systèmes informatiques de gestion de l'information, en axant leur conception sur les utilisateurs. A la base, les logiciels de reconnaissance vocale servent surtout à entrer du texte en masse tout en se passant du clavier (qui offre un débit de 50 mots par minute contre plus de 150 pour la parole), le clavier reste cependant encore nécessaire aux corrections de texte et à l'utilisation de l'ordinateur.

Nous allons abordé dans ce PPE une technique de reconnaissance vocale basée sur la reconnaissance de mots isolés dans le cas de petit vocabulaire. Il s'agit d'une technique s'appuyant sur les coefficients cepstraux dans l'échelle des Mels (Mel Frequency Cepstrum Coefficients, MFCC). Cette technique bien que très limitée est largement utilisée dans la reconnaissance vocale à petit dictionnaire.

A partir de là nous voyons deux approches possibles de la reconnaissance vocale :

- **Global :** Ce type de reconnaissance vocale concerne la reconnaissance de mots isolés. Il s'agit le plus souvent de système mono locuteur avec phase d'apprentissage avec création d'un dictionnaire. Le mot est l'entité élémentaire de la reconnaissance vocale, c'est ce type de reconnaissance vocale qui sera utilisé dans ce PPE.

- **Analytique :** Ici on considère la parole comme un ensemble de phonèmes ce qui implique l'utilisation de l'intelligence artificielle. Le signal passe par une étape appelée Décodage Acoustique Phonétique (DAP) où le signal est transcrit en une suite de phonème à analyser. Des analyses syntaxiques et grammaticales permettent de retirer l'ensemble des phrases incorrectes ou les orthographes incorrectes. Cette méthode est très lourde mais apporte des

résultats très satisfaisants. Cette approche est utilisée dans les applications de dictée vocale comme par exemple avec ViaVoice d'IBM.

b) L'échelle des Mels.

L'échelle des Mels est une échelle biologique. C'est une modélisation de l'oreille humaine. A noter que le cerveau effectue en quelque sorte une reconnaissance vocale complexe avec filtrage des sons… Prenons l'exemple suivant où vous êtes à table en compagnie de nombreuses personnes, l'ensemble de ses personnes parle en même temps et vous discutez avec votre voisin. Malgré le bruit, vous arrivez à discerner clairement ce que vous dit votre voisin, vous ignorez de façon naturelle le bruit de fond et vous amplifier le son qui vous paraît le plus important. Vous pouvez répéter cette expérience avec chacun des convives. Le cerveau ne se contente non pas seulement de filtrer les sons et de les amplifier mais aussi de prédire. Prenons l'exemple suivant où une personne discute avec vous avec un volume sonore très bas, vous n'avez pas entendue une certaine partie de la phrase mais vous arrivez à la reconstituer et à la comprendre.

A partir de l'étude du cerveau nous pouvons nous faire une idée de la complexité de la reconnaissance vocale et nous pouvons nous rapprocher d'un modèle de plus en plus puissant et parfait.

On considère que l'oreille humaine perçoit linéairement le son jusqu'à 1000 Hz, mais après, elle perçoit moins d'une octave par doublement de fréquence. L'échelle de Mels modélise assez fidèlement la perception de l'oreille : linéairement jusqu'à 1000 Hz, puis logarithmiquement au dessus.

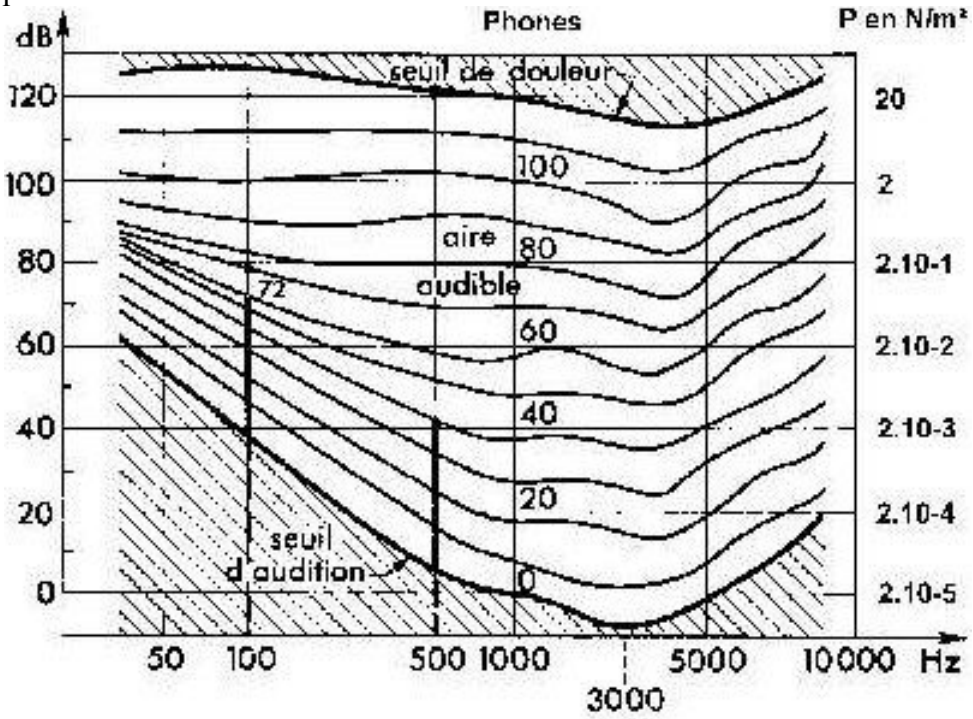
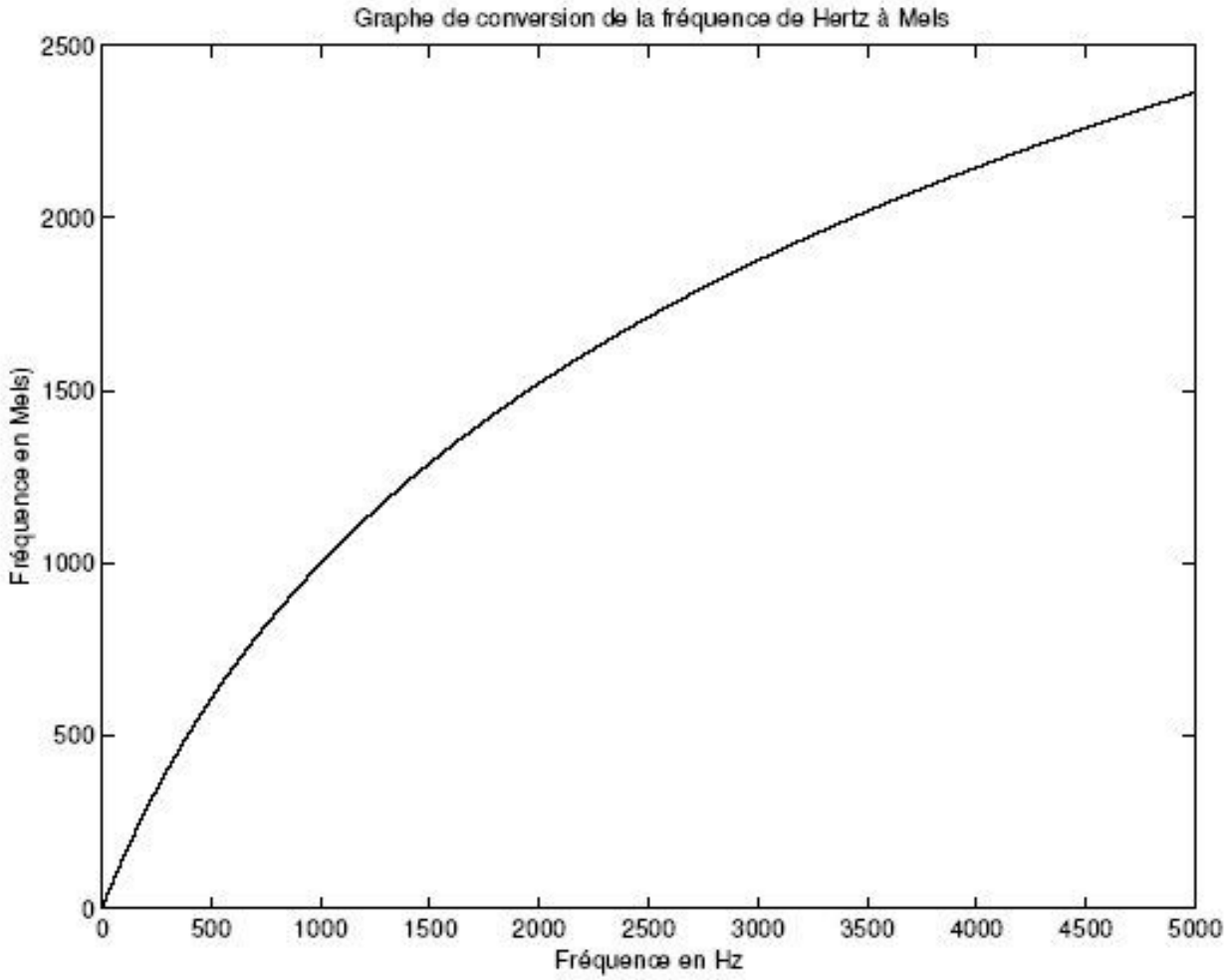


Diagramme de Fletcher : Décrit les seuils de perception et lignes d'isotonies selon les fréquences. On voit bien que la perception n'est pas linéaire.

La formule donnant la fréquence en Mels m à partir de celle en Hz f est :

$$m = \frac{1000 \cdot \ln\left(1 + \frac{f}{700}\right)}{\ln\left(1 + \frac{1000}{700}\right)} \approx 1127 \cdot \ln\left(1 + \frac{f}{700}\right) \approx 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)$$

L'échelle des Mels permet donc de modéliser une perception de l'oreille linéairement.



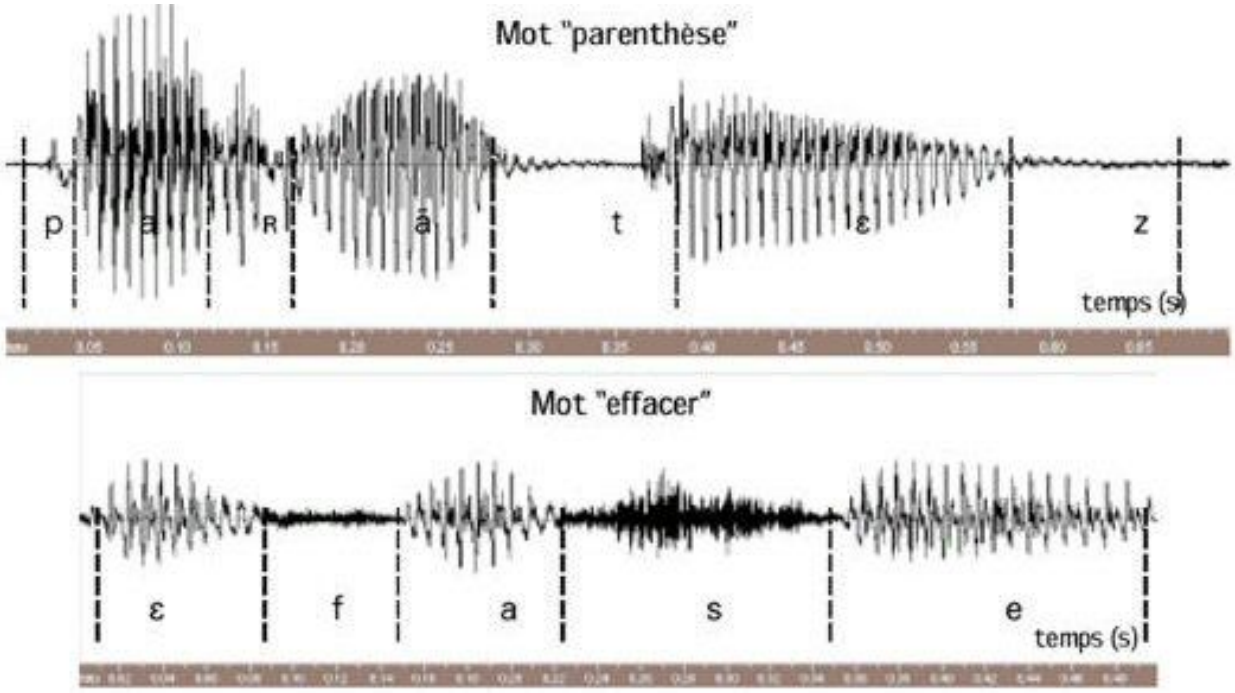
On remarque qu'avant 1000 Hz, la courbe est à peu près droite, ce qui traduit bien l'équivalence entre Hz et Mels à ces fréquences.

Exemple : L'octave d'un son de 2000 Hz (1800 Mels) sonnera l'octave supérieure à 4600 Hz (3600 Mels) au lieu de 4000 Hz.

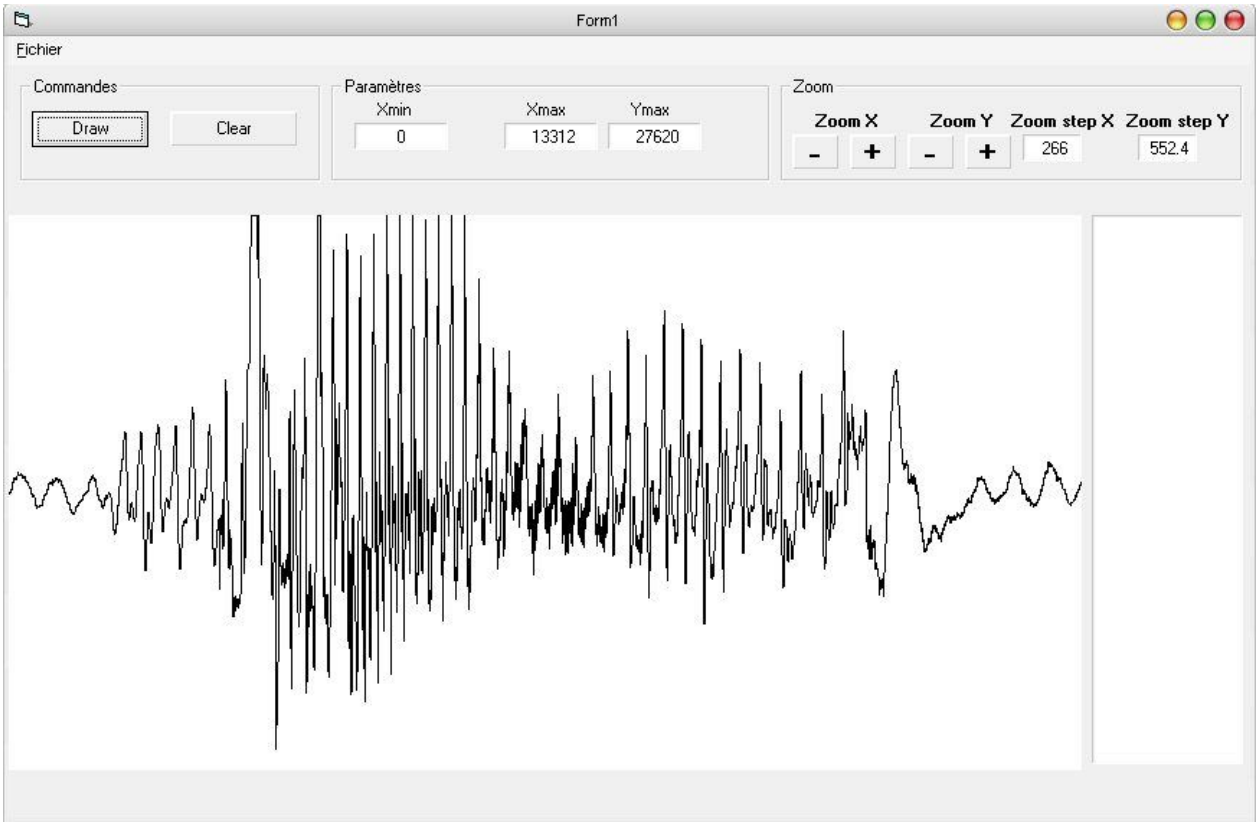
V- Extraction des paramètres.

La parole étant un signal constitué d'une infinité d'informations, il faut en extraire les informations les plus importantes.

Exemple avec les signales audio de 2 mots :

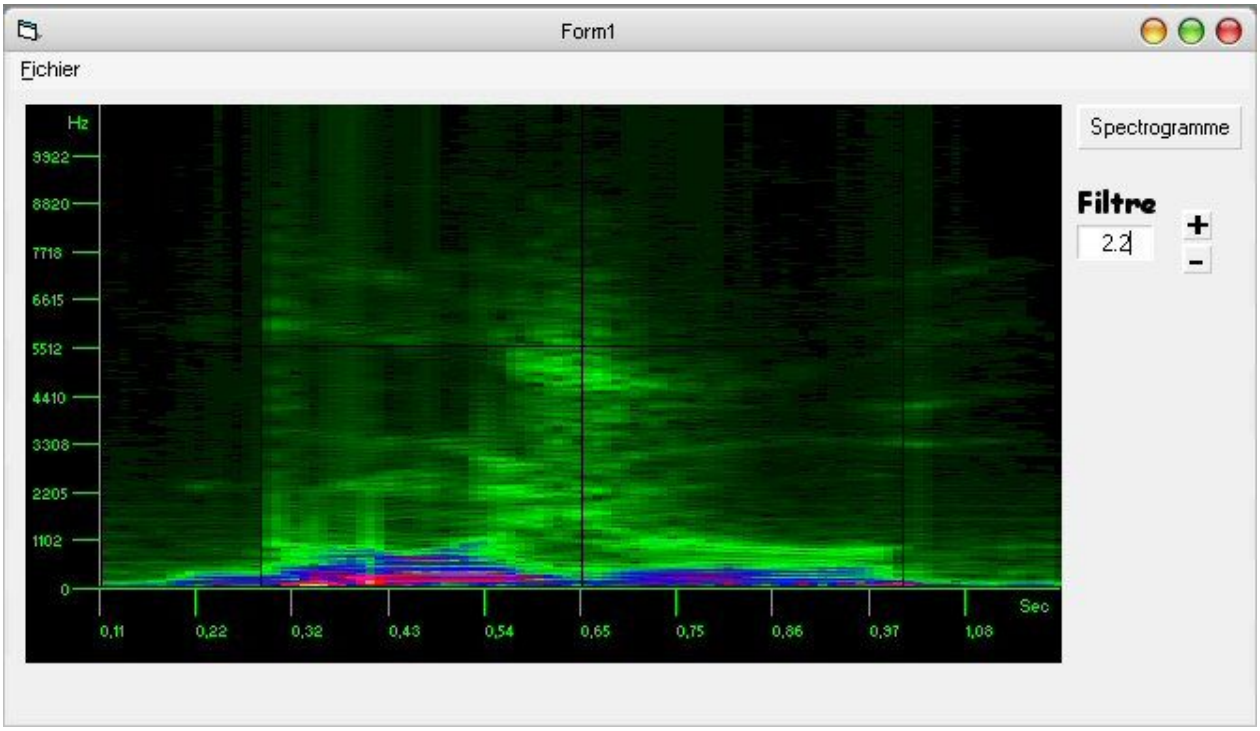


Un traitement direct de comparaison sur ce genre de signal est impossible car il y a trop d'informations et surtout ses informations ne sont pas exploitables.



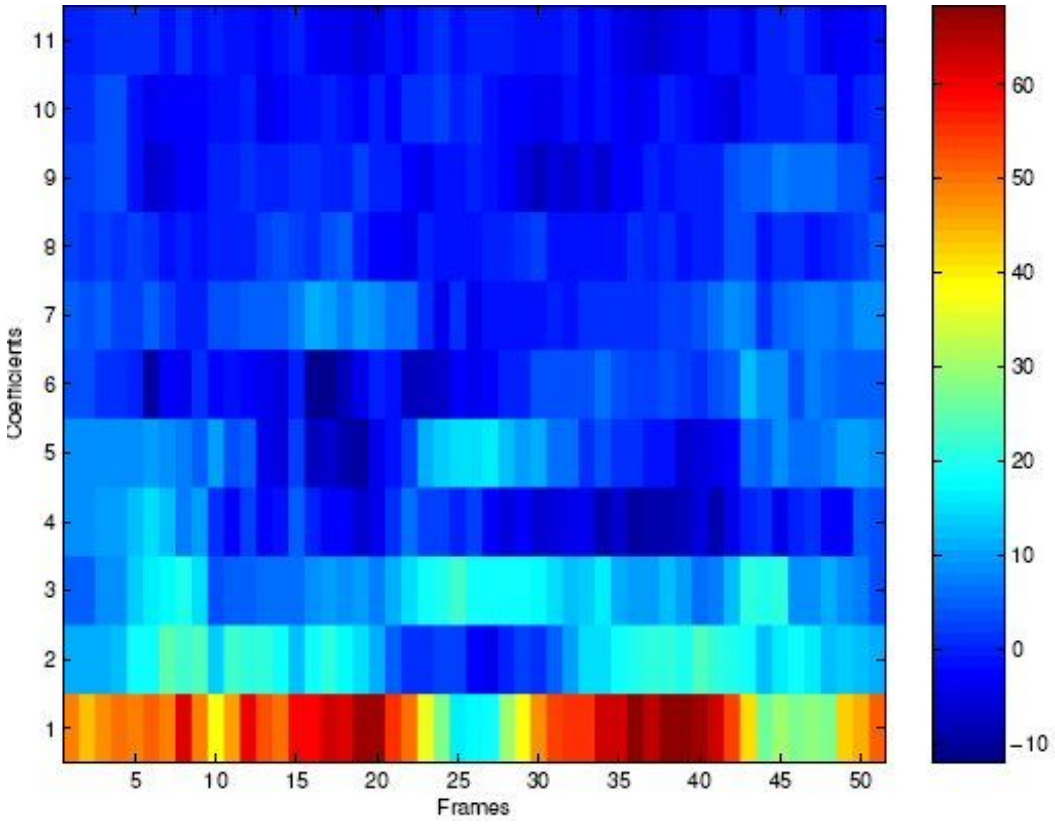
Signal audio du mot ‘Bonjour’ obtenu avec le programme DrawPts de notre PPE réalisé avec Visual Basic 6.0.

Dans les années 50, un appareil appelé Sonagraphe fit son apparition sur le marché. Il permettait de faire un spectrogramme d'un signal. C'était un appareil analogique à aiguilles, il est remplacé aujourd'hui par des traitements numériques utilisant les transformées de Fourier.



Spectrogramme du mot 'Bonjour' obtenu avec le programme DrawSpectre de notre PPE réalisé avec Visual Basic 6.0.

A partir des spectrogrammes, on pourrait se contenter d'un simple calcul de distance spectrale entre le spectre du mot à reconnaître et celui du mot présent dans le dictionnaire, mais le fait qu'il y a des variations inévitables dans la prononciation nécessite l'utilisation d'un algorithme de comparaison dynamique. La comparaison dynamique directe de deux spectres donne des résultats peu convainquant et le temps de calcul est assez élevé (dû au fait qu'il y a encore beaucoup de données à traiter). C'est pour cela que l'on a recourt à d'autres techniques, dont les coefficients cepstraux dans l'échelle des Mels (MFCC). Il s'agit tout simplement d'extraire d'un spectre les informations les plus pertinentes, sur un nombre fini de coefficients en fonction du temps. Dans notre PPE nous avons utilisé 12 coefficients.

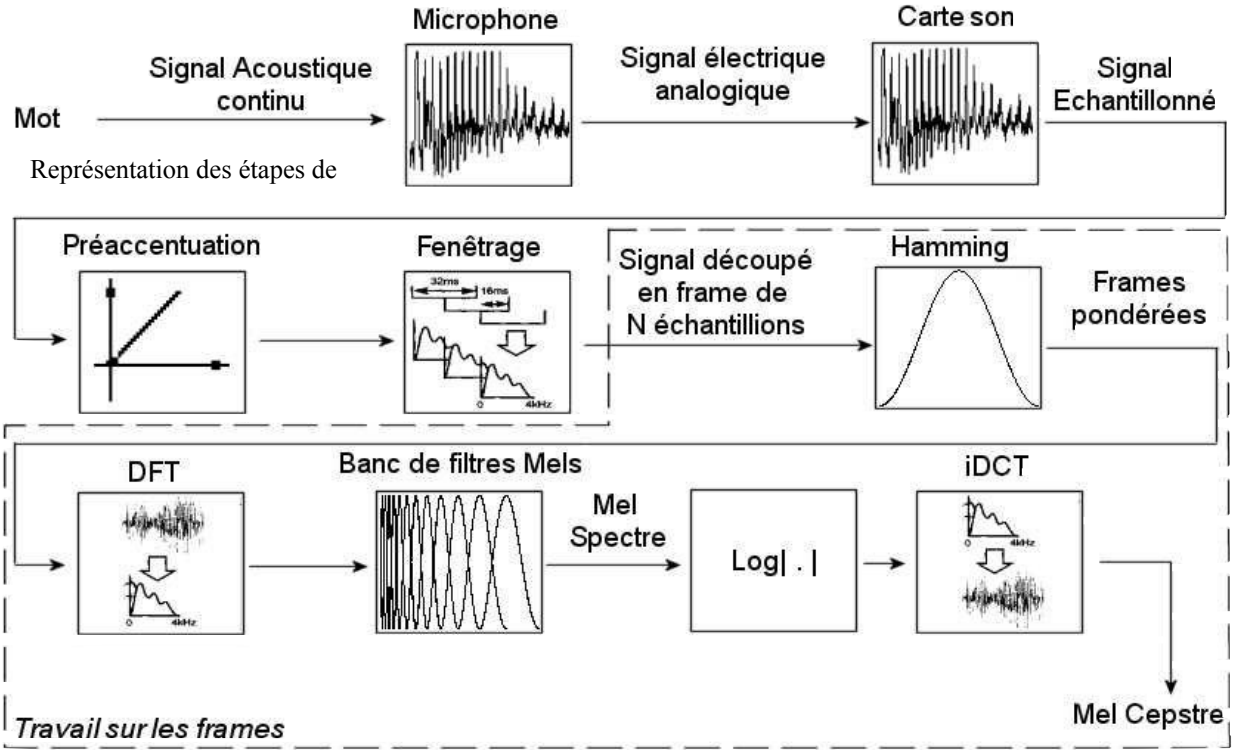


Représentation des MFCC du signal précédent ('Bonjour'). Il y a beaucoup moins d'informations à traiter. Concrètement il s'agit d'une matrice 12x51. (12 Filtres sur 51 fenêtres audio).

VI- Création d'un cesptre à partir des coefficients MFCC (Mels Frequency Coefficients Cepstrum).

a) Introduction.

Notre reconnaissance vocale fonctionne à partir de comparaison entre cepstres MFCC. Plusieurs étapes sont nécessaires pour transformer un fichier audio en cepstre MFCC. On détaille dans cette section la construction pas à pas de ces coefficients. Le principal intérêt des MFCC est d'extraire des informations pertinentes et en nombres limités en s'appuyant à la fois sur la production théorique (théorie cepstrale) et à la fois sur la perception de la parole (échelle des Mels).



Représentation des étapes de création des MFCC.

b) Prétraitement.

On considère à partir de maintenant que le signal d'un mot a été numérisé (ou échantillonné). Il est représenté à partir de maintenant par une famille $(Xn)n \in [1, M]$ où M est le nombre d'échantillons dans le signal. Chaque élément de la famille est un réel compris entre -1 et 1 (en effet, pour un codage sur 16 Bits, les valeurs sont codées sur 15 Bits, plus un bit pour le signe, ce qui donne des valeurs possibles entières entre -32768 et 32767, ramenée à des valeurs réelles entre -1 et 1). Le signal est échantillonné (par la carte son sur un ordinateur) à une

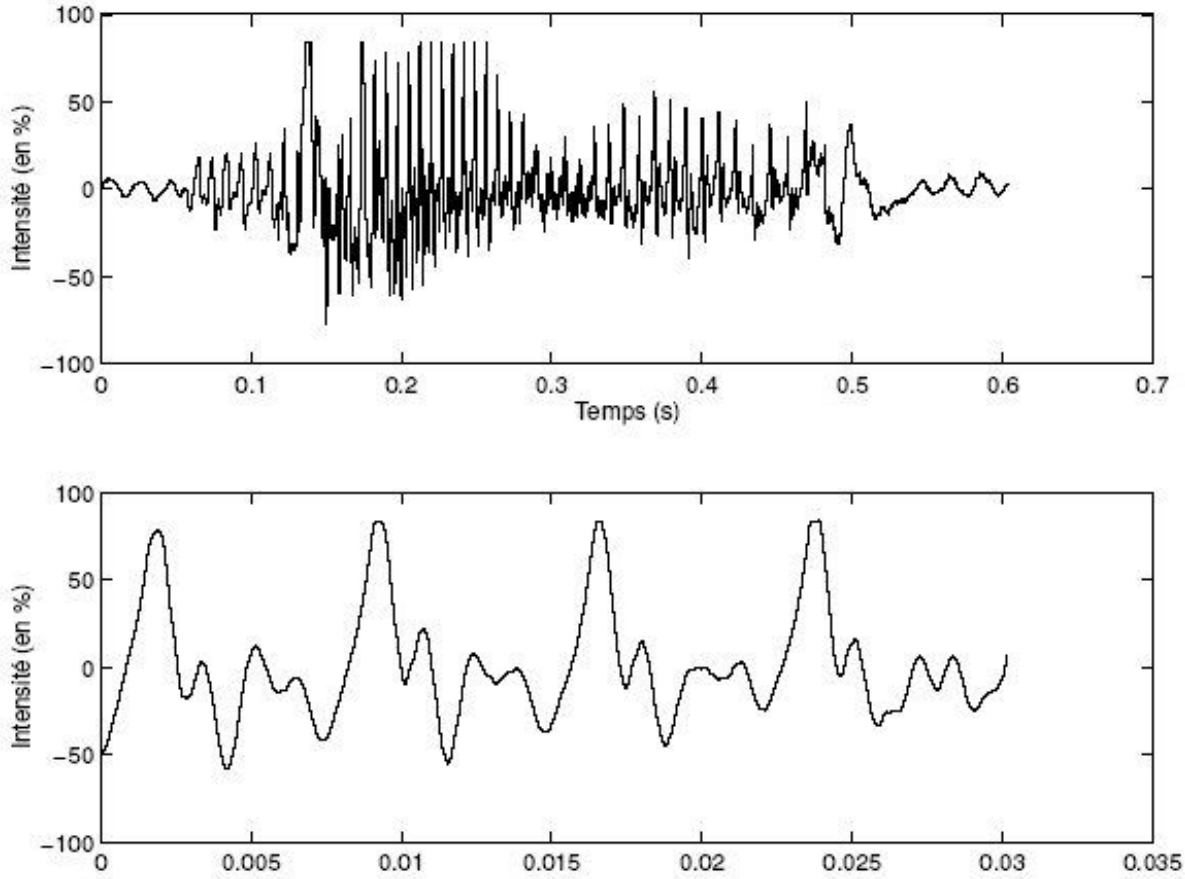
fréquence *Fs* (communément 11 025 ou 22 050 Hz), c'est-à-dire que en une seconde, 11 025 ou 22 050 points on été créé.

On considérera par la suite que le signal de la parole est délimité de manière à bien commencer par le début d'un mot, qu'il se termine sans trop de blanc par la fin de ce mot et qu'il n'y a pas de parasite (enregistrement idéal sans bruit en laboratoire).

A noter que notre programme principale principal nommé 'engine.exe' utilise des fichiers audio mono sur 16 Bits échantillonnés sur 22 050 Hz.

c) La parole en tant que signal quasi-stationnaire.

La parole est un son complexe dans son ensemble, mais si on considère des intervalles de temps très réduits (environ 20 ms), le signal vocale est alors presque stationnaire. C'est alors pour cela que l'on considère le signal vocal comme un signal quasi-stationnaire, en première approximation. On pourra utiliser des fonctions de fenêtrage pour améliorer les résultats. Il est à noter aussi qu'un signal audio ne peut pas être traité dans son ensemble car cela demanderait énormément de calculs pour la machine (impossible à réaliser), ainsi on verra que l'on découpe le signal audio en fenêtre (par paquet de 2^n échantillons).



Représentation temporelle du mot 'Bonjour' (au dessus) ainsi qu'un extrait de 30 ms pendant la prononciation de la voyelle [on] (en dessous). On remarque alors que le signal du bas peut être considéré comme un signal périodique. Ainsi on cherchera ici à extraire les informations pertinentes.

d) Fenêtrage.

Les sites traitant de la reconnaissance vocale par MFCC explicites clairement que le signal doit tout d'abord suivre une préaccentuation pour palier le fait que les hautes fréquences seront moins puissantes que les basses fréquences. Or d'après nos tests dans notre programme principal, le fait de préaccentuer un signal le détruit complètement. Nous n'avons donc pas introduit cette étape dans ce dossier.

La formule de préaccentuation d'un signal est de la forme : $h_n = 1 - \alpha . Z_n^{-1}$

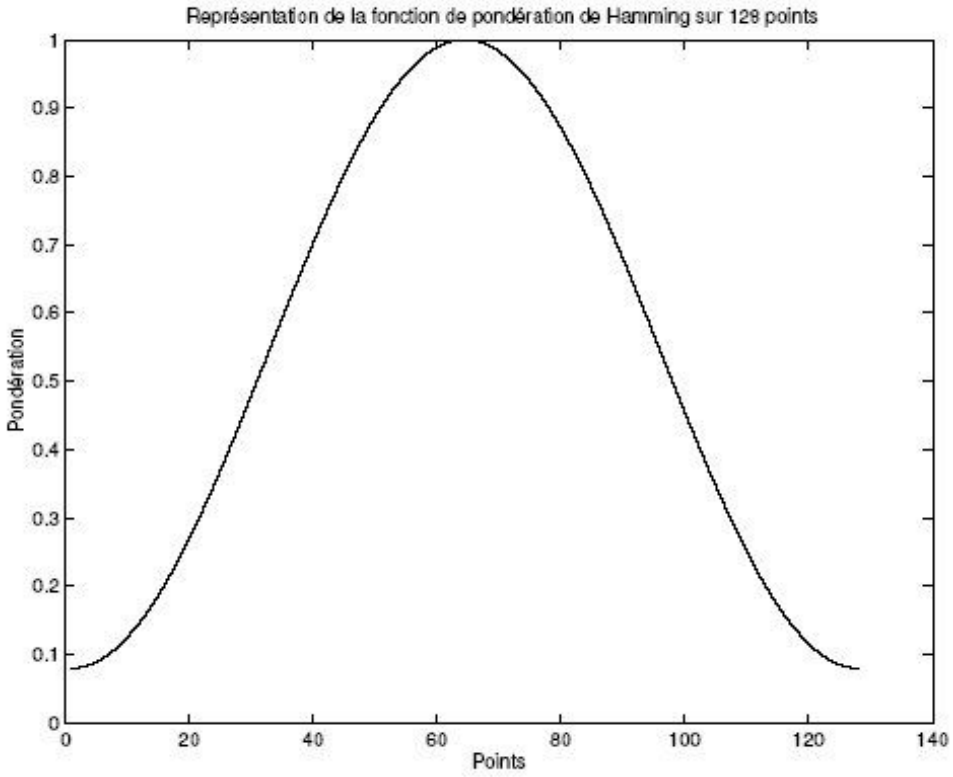
Avec α le facteur de préaccentuation, pris communément à 0.970.

Une fois la formule de préaccentuation du signal effectuée, nous découpons le signal en fenêtres. Le signal est découpé en tranches de 2^n échantillons appelées frames ou encore fenêtres qui ont la particularité de se recouvrir de moitié dans l'objectif d'avoir un meilleur traitement pour FFT (Fast Fourier Transform). On utilise typiquement une fenêtre de $N = 512$ échantillons ou un nombre qui est une puissance de 2, cela vient du fait que l'algorithme FFT que nous utilisons est bien plus rapide pour ces nombres. Dans notre programme principal nous utilisons des fenêtres de 512 échantillons.

e) Application d'une fenêtre de pondération.

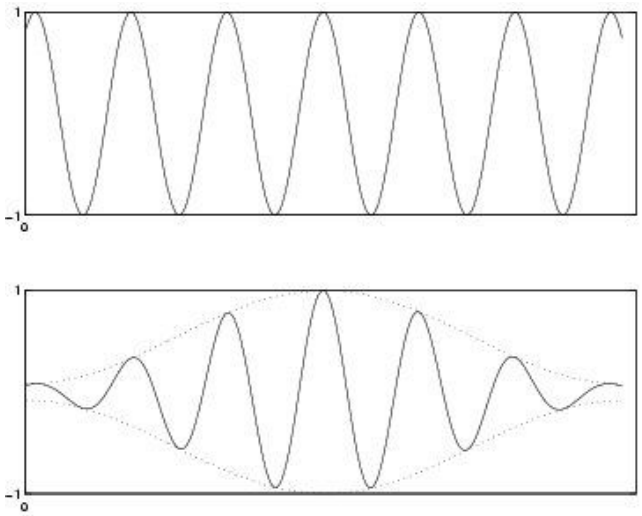
Une fenêtre de pondération est appliquée à chaque frame ceci dans l'objectif d'harmoniser les échantillons pour permettre un meilleur traitement pour l'algorithme FFT. En effet FFT ne donne pas de bons résultats quant une pente trop importantes est détectée dans une partir du signal. La fenêtre de pondération a pour objectif de minimiser les erreurs produites par FFT.

Fenêtre de pondération Hamming : $0.54 - 0.46.\cos\left(\frac{2\pi.n}{N-1}\right), n \in [0, N-1]$



Courbe de la fonction de pondération de Hamming sur 128 points.

Avantages et inconvénients du fenêtrage par Hamming : améliore la netteté des harmoniques et supprime les discontinuités sur les bords (effets de bord), mais on perd une information sur l'intensité des pics harmonique.



Représentation d'un signal sinusoïdale non pondéré puis pondéré par la fenêtre de Hamming.

f) DFT (Discret Fourier Transform) ou encore FFT (Fast Fourier Transform).

Joseph FOURIER (1768- 1830) Physicien – Mathématicien, modélise la propagation de la chaleur en inventant la fameuse série trigonométriques qui permettent d'exprimer des fonctions discontinues comme somme d'une série infinie de sinus et de cosinus. Il a démontré que tout signal pouvait se décomposer en sommes de fonctions sinus élémentaires La transformée de Fourier rapide (Fast Fourier Transform - FFT) a été introduite en 1960. C'est un algorithme très puissant mais qui possède la contrainte de ne pouvoir utiliser que des séries de 2^n nombres.

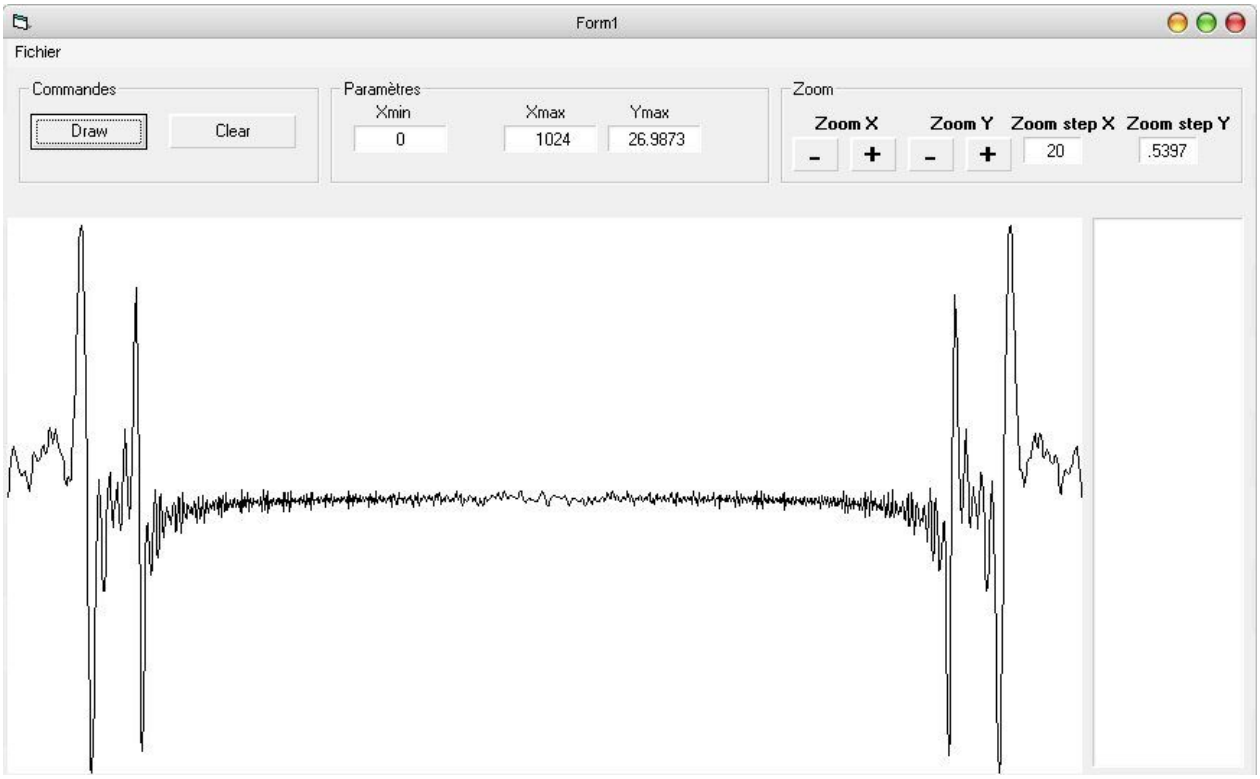
Cette étape consiste à prendre chaque fenêtre et à appliquer la transformée de Fourier (on utilise dans notre programme une implantation de Fourier à base de cosinus adaptée à des signaux audio dans l'objectif d'optimiser notre programme), on convertit ainsi chaque frame du domaine temporel en domaine fréquentiel. La DFT sur la $i^{ème}$ frame est définie ainsi :

$$Y_{i,n} = \sum_{k=1}^{N} y_{i,k} \exp\left(\frac{-2\pi j(n-1)(k-1)}{N}\right), \ avec \ n \in [1,N] \ et \ j^2 = -1 [10,9]$$

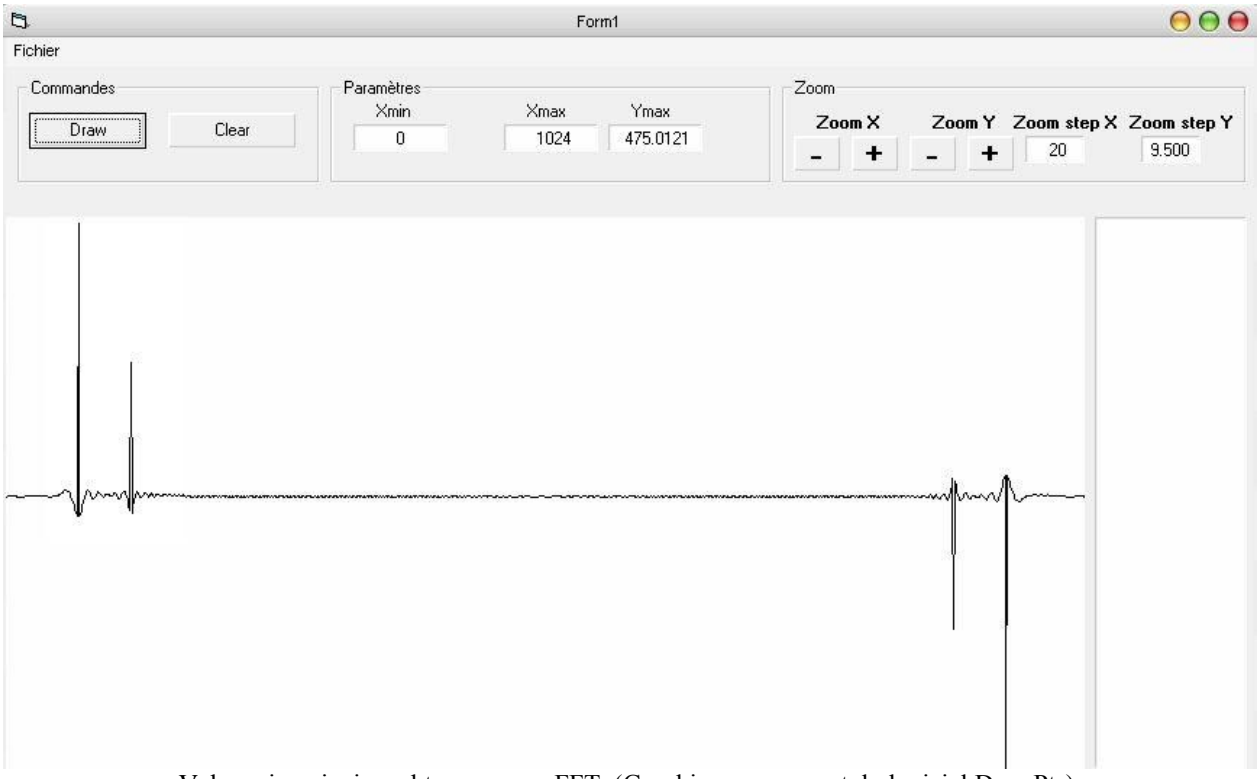
Signal simple sur 128 points sans bruit somme de deux signaux :

$A0.\sin(2\pi.F0.t) + A1.\sin(2\pi.F1.t)$ *avec* $A0$, $A1$ *les amplitudes et* $F0$, $F1$ *les fréquences des signaux.*
Ici A0=1, A1=A0/2, F0=150 Hz et F1=250 Hz.

Application de FFT sur le signal du dessus, FFT sur 1024 points.



Valeurs réelles obtenues avec FFT. (Graphique provenant du logiciel DrawPts).



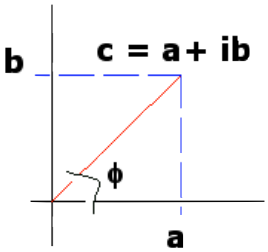
Valeurs imaginaires obtenues avec FFT. (Graphique provenant du logiciel DrawPts).

Voilà ce que donne la transformée de Fourier sur une frame, ici une frame de 1024 points de notre somme de sinus. Nous obtenons les valeurs réelles et imaginaires données par les nombres complexes donnés par FFT.

Avant de transformer les nombres complexes en amplitudes et en fréquences compréhensibles, nous devons faire une mise à l'échelle. En effet l'axe Y des ordonnées est incompréhensible. Nous avons dit au début qu'un signal audio a des amplitudes comprises entre -1 et 1. L'axe X des abscisses représente pour chaque point une fréquence !

Ensuite nous remarquons que FFT donne deux fois les mêmes informations, en effet sur 512 points on retrouver de façon inversée la même chose sur les 512 points suivants. On va donc supprimer les 512 points qui ne nous intéressent pas !

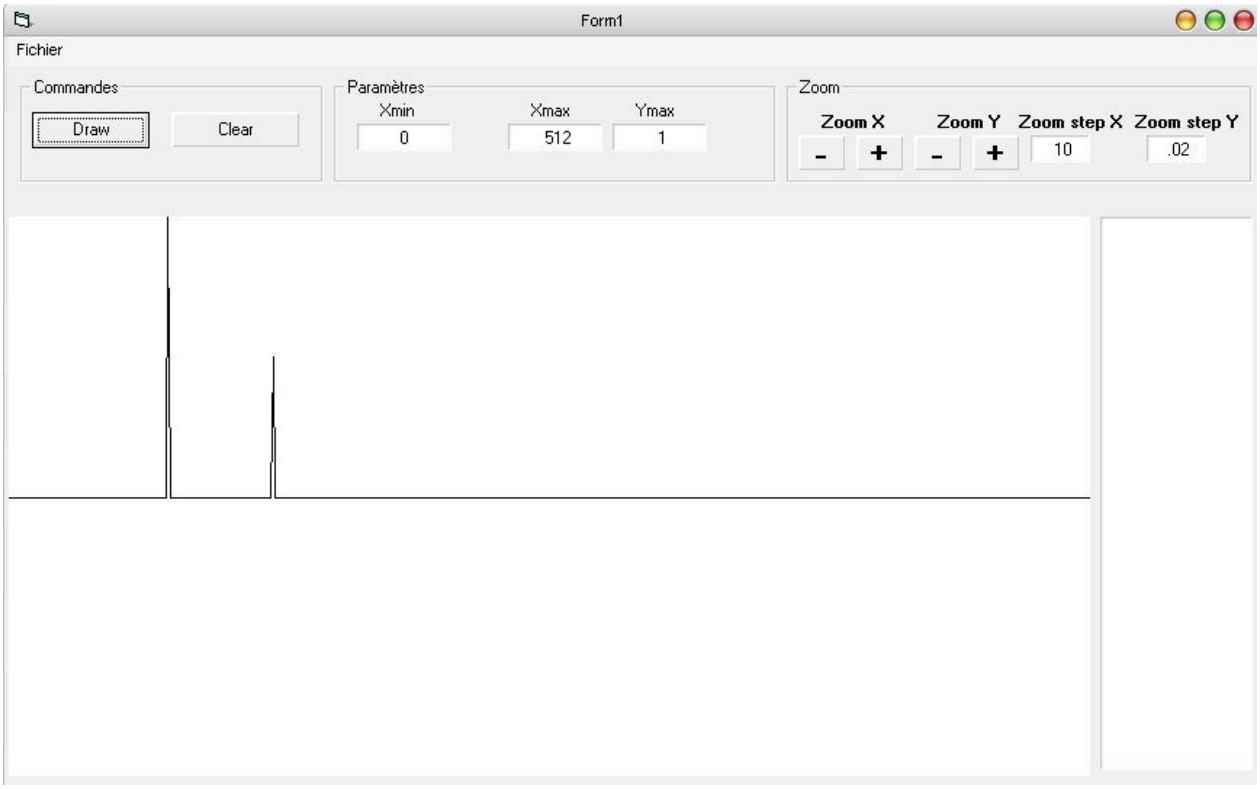
FFT donne énormément d'informations, les 2 graphiques précédents peuvent paraître déroutant aux premiers abords, dans notre cas constituer un spectre d'amplitudes des fréquences dans le temps nous suffit largement. Pour cela il suffit de calculer le module de chaque nombre complexe.



Pour cela on applique la formule suivante en rapport à l'image ci-contre :

$r = \sqrt{a^2 + b^2}$ *avec a les valeurs réelles et b les valeurs complexes.*

Nous obtenons le graphique suivant :



Amplitudes en fonction des fréquences obtenue par calcul des nombres complexes provenant de FFT. (Graphique provenant du logiciel DrawPts).

Notre programme *DrawPts* ne donne pas de graduations mais on constate plusieurs choses à partir de ce graphique :

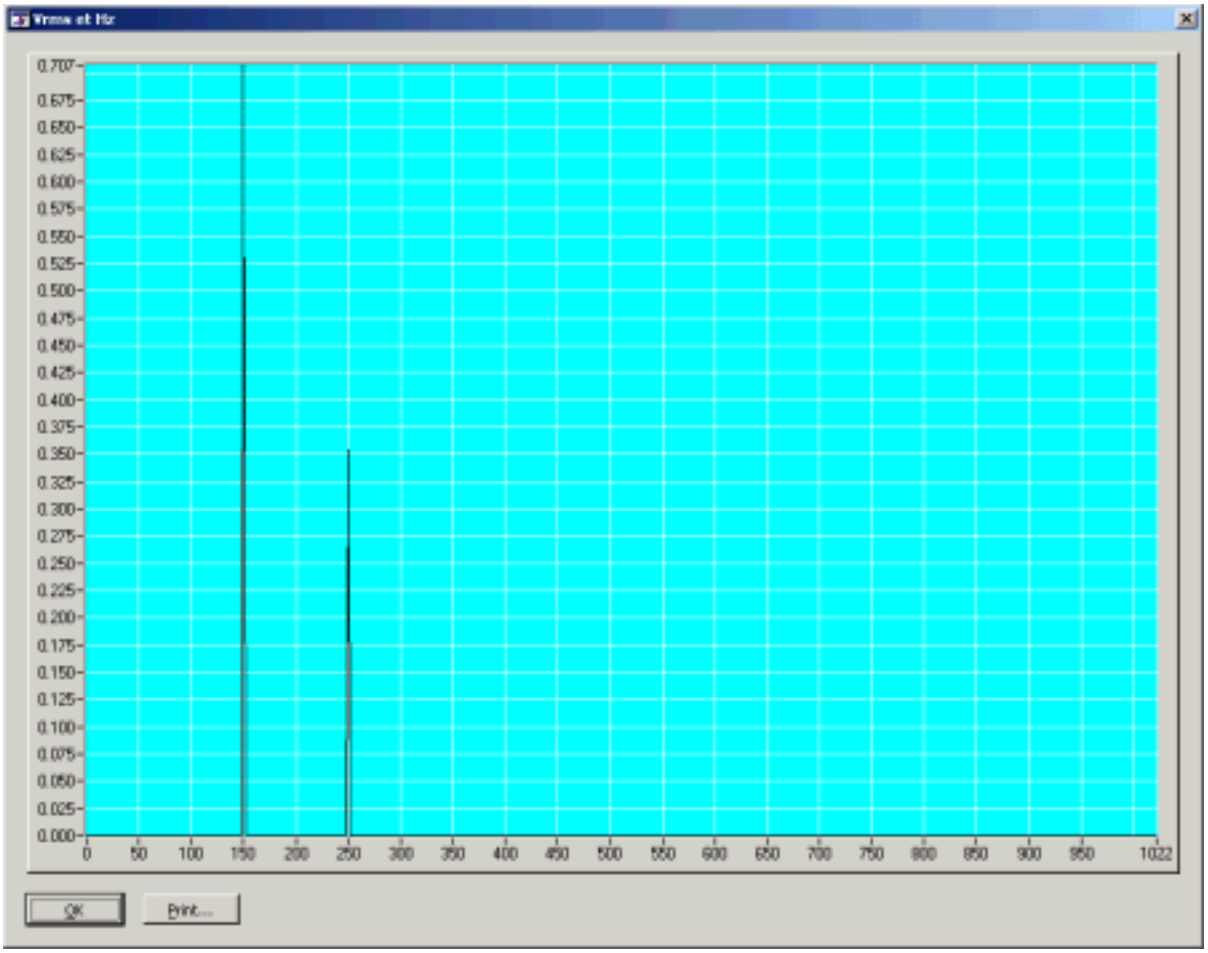
- Notre signal de départ est la somme de 2 signaux.
- Les amplitudes sont situées entre 0 et 1.
- Si on s'attarde sur l'axe X des abscisses, nous retrouvons les fréquences.

Sachant que nous avons notre signal d'origine composé de deux sinusoïdales sur 1024 point soit on peut dire échantillonné à 1024 Hz, la plus grande fréquence que nous pouvons obtenir sera de 512 Hz. En effet le théorème d'échantillonnage de Nyquist-Shannon dit que « la fréquence d'échantillonnage d'un signal doit être égale ou supérieure à deux fois la fréquence maximale contenue dans ce signal, afin de convertir ce signal d'une forme analogique à une forme numérique ». Ce théorème est à la base de la conversion numérique des signaux. Par exemple l'oreille humaine pouvant capter les sons jusqu'à 22 KHz, il convient, lors de la conversion, d'échantillonner les fréquences sonores jusqu'à 44 KHz.

On montre que si F est la fréquence d'échantillonnage du signal et que N est le nombre d'échantillons alors la résolution (le pas entre deux graduations sur l'axe des fréquences) vaut :

$$\Delta F = \frac{F}{N}$$

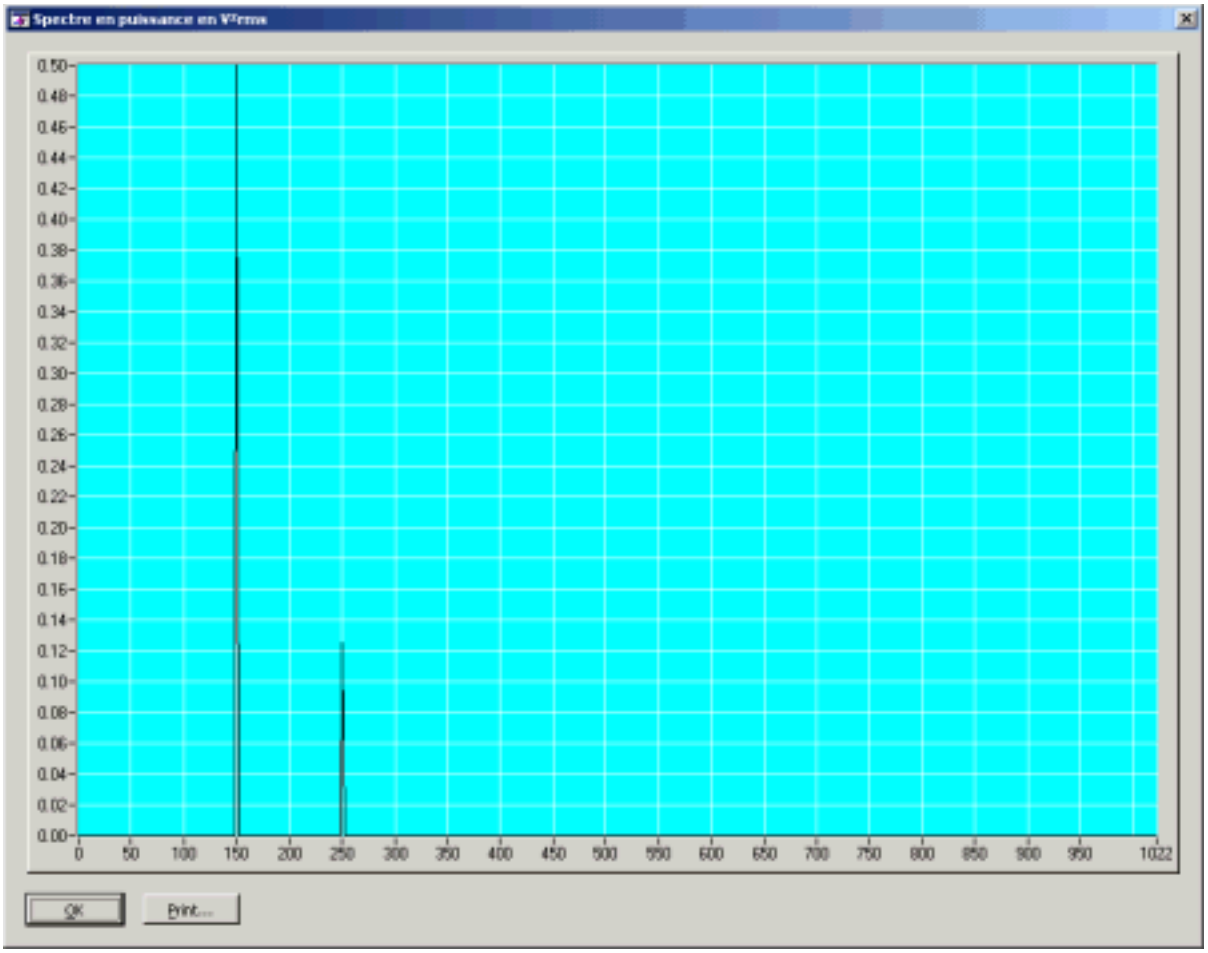
Notre pas sera donc de 1 Hz pour chaque point du graphique, on obtient ainsi :



L'axe des X est gradué en fréquence

On devine maintenant les fréquences de nos deux signaux soient 150 Hz et 250 Hz grâce à un bon changement d'échelle, mais ce n'est pas pour autant fini. En effet, il est souvent intéressant de connaître la puissance d'un signal selon ses différentes harmoniques. Pour cela on utilise directement la formule suivante :

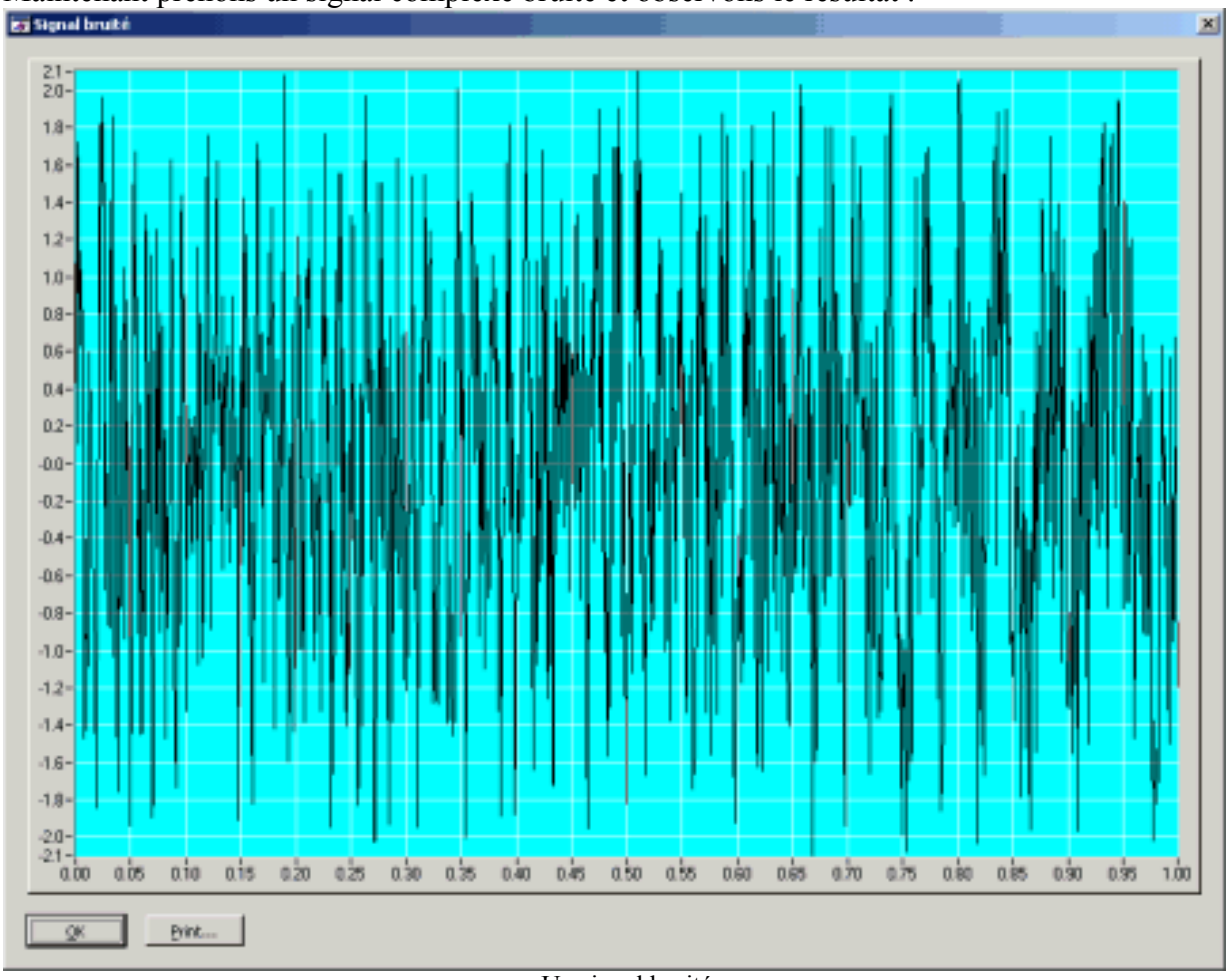
$P_i = \frac{\sqrt{a^2 + b^2}}{N^2}$ *où N représente le nombre d'échantillons et a et les composantes complexes de FFT.*



Spectre en puissance

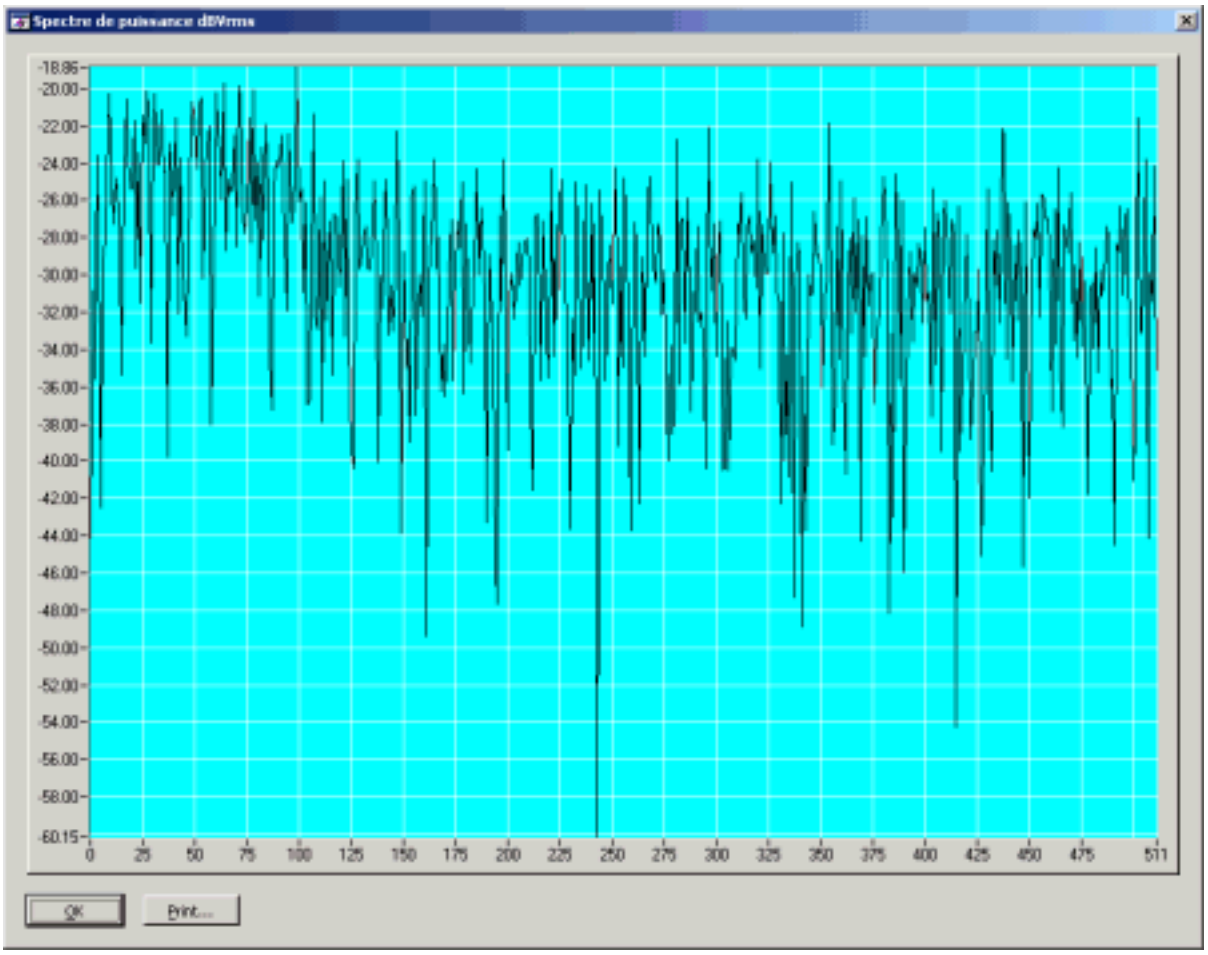
On observe que le signal dissipe 0.5 Watt à 150 Hz et 0.12 W à 250 Hz.
Nous avons ainsi notre spectre !

Maintenant prenons un signal complexe bruité et observons le résultat :



Un signal bruité.

On "sait" que dans cet amas, il y a quelque chose alors on se dit que, pas de problème, avec une bonne FFT, on va s'en sortir. Que nenni... La preuve :

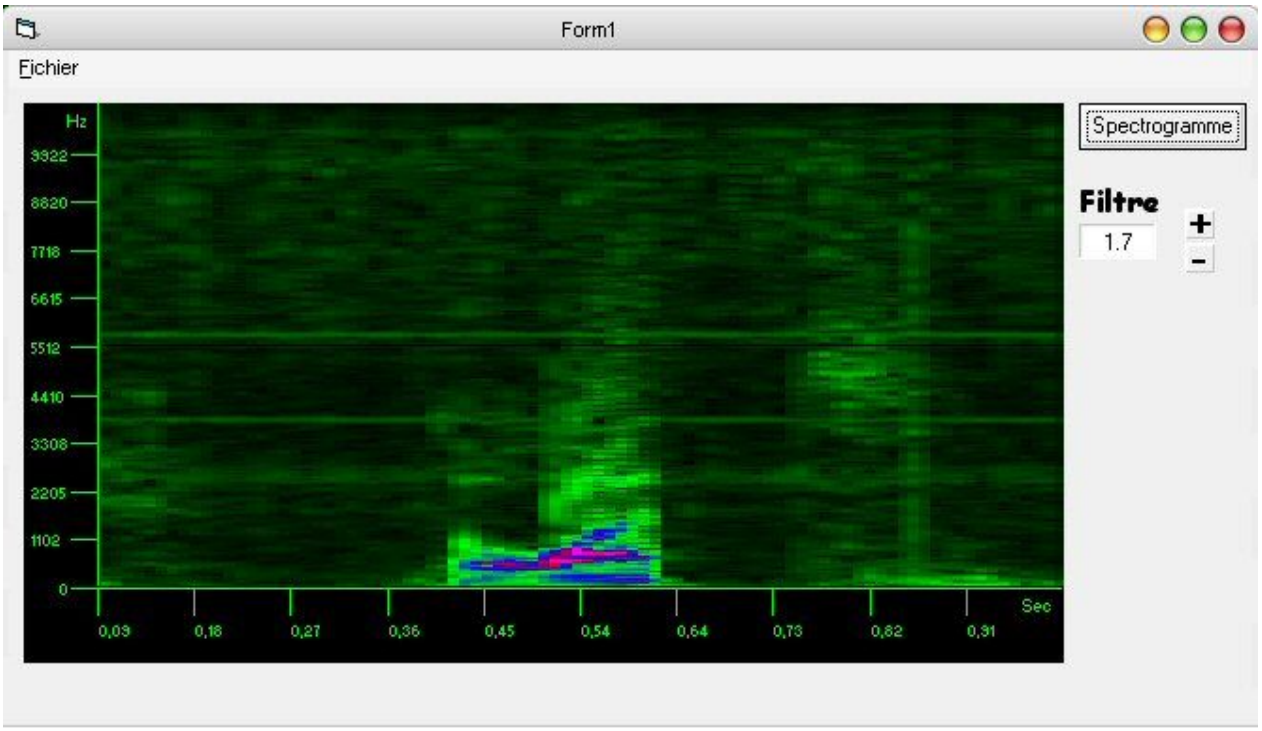


Spectre de puissance du signal bruité précédent.

En effet quand on regarde le spectre du signal en question, il est impossible de retrouver une information quelconque.

Mais une solution existe et consiste à découper notre signal en frames (fenêtres), comme nous l'avons expliqué, appliquée une FFT sur chaque fenêtre et représenté en 3D sur chaque intervalle notre spectre de puissance. Un spectre en 3D se présente de cette façon, l'axe X représentant l'axe des temps, l'axe Y, l'axe des fréquences et enfin l'intensité des couleurs représentant l'amplitude des spectres.

On va prendre l'exemple d'un son court parasité représentant le mot « droite », le spectre suivant nous permet de bien traiter l'ensemble des informations généré par FFT :



Spectre du mot « droite » obtenu avec notre logiciel DrawSpectre.

g) Conclusion sur le prétraitement du signal et la transformée de Fourier.

Nous avons vu jusqu'ici comment créer un spectre à partir d'un signal audio brut, c'est-à-dire passer du domaine temporel au domaine fréquentiel. Nous avons dû auparavant effectuer une série de traitements sur le signal pour améliorer les résultats fournis par FFT, plus particulièrement, affiné notre spectre.

Il faut aussi savoir que la transformée de Fourier et réversible, il est ainsi possible de partir d'un spectre pour recréer un signal audio identique à celui d'origine ! Pour cela on applique la transformée inverse de Fourier. La transformée de Fourier est utilisé dans de nombreux domaines, tant bien dans la compression de données numériques (Normes MP3, MPEG…), dans le domaine médicale avec l'étude du cerveau, l'aéronautique… Plus amples informations sur la transformée de Fourier ici : http://perso.wanadoo.fr/philippe.baucour/pratiquer/fft/fft1.html

Nous en sommes au stade où une simple comparaison entre deux spectres serait possible et nous donnerait l'illusion d'avoir enfin terminé, c'est-à-dire, comparé un mot avec un dictionnaire et le reconnaître. Or les spectres que nous obtenons contiennent trop d'informations

et de plus nous ne prononçons pas de façon identique les mêmes mots ; il existe toujours des variations d'intensités de voix et des variations temporelles de prononciation. Nous allons donc devoir changer d'échelle et ainsi transformer nos spectres en cepstres dans l'échelle des Mels. C'est à partir d'ici qu'interviennent les MFCC. (Mel Frequency Cepstrum Coefficients).

h) Banc de filtres Mels.

Parce que l'étendue des fréquences présentes dans le spectre est encore très large, donc beaucoup de données à traiter, on a recours au banc de filtres dans l'échelle de Mels. On relie ainsi le système de reconnaissance vocale au fonctionnement de l'oreille humaine. Il s'agit de filtres passes bandes (fonction fenêtre de Hamming) centrés linéairement dans le domaine fréquentiel des Mels et de largeur telle qu'ils divisent l'espace des fréquences de manière égale dans le domaine des Mels et qu'ils se recouvrent chacun par moitié. Les filtres sont donc disposés logarithmiquement dans l'échelle des fréquences usuelles (Hz), on a ainsi beaucoup de filtres pour les basses fréquences alors que les hautes fréquences sont disposées plus largement.

On rappel la formule donnant la fréquence en Mels à partir de la fréquence en Hz :

$$m = 2595.\log_{10}\left(1+\frac{f}{700}\right) \; avec \;\; f \;\; la \;\; fréquence \;\; en \;\; Hertz.$$

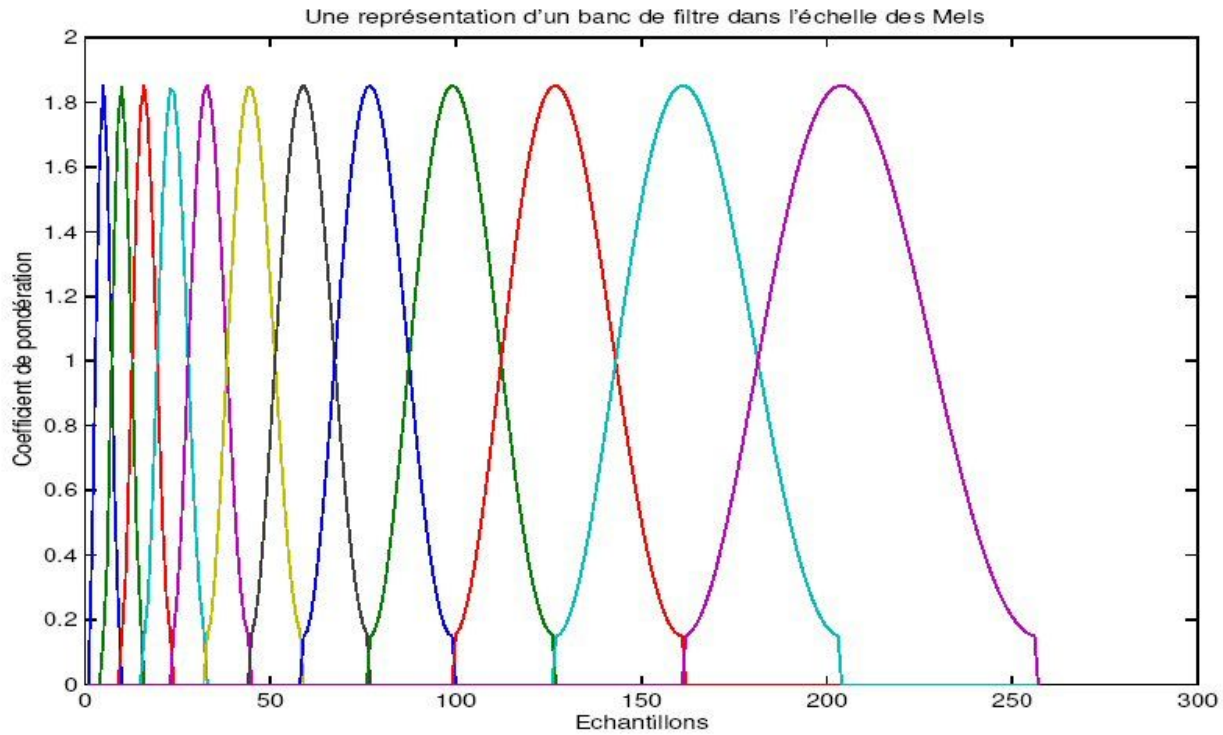
Chaque filtre va donner un coefficient cepstrale :

$$S_{i,k} = \sum_{n=0}^{N/2} Y_{i,n} M_{n,k}, \;\; k = 0...K$$

Avec : $Y_{i,n}$ le $n^{ème} \in [1,N]$ coefficient de la transformée de la $i^{ème} \in [1,I]$ frame, et $M_{n,k}$ le $n^{ème} \in [1,N]$ coefficient du $k^{ème} \in [1,K]$ filtre. On utilise communément 12 coefficients, on utilise alors K=13 filtres (pour obtenir 12 coefficients, il faut un filtre de plus car le $0^{ème}$ est inutile).

On a donc $S_{i,k}$ la matrice de sortie du $k^{ème}$ filtre pour la $i^{ème}$ frame. On a, à cette étape, ce qu'on appelle un Spectre Mel (Spectrum Mel).

Une représentation d'un banc de filtres dans l'échelle des Mels :



Représentation de la matrice *N* x *K* des coefficients du banc de filtres (*N* : Nombre d'échantillons par frames, *K* nombres de filtres souhaités).

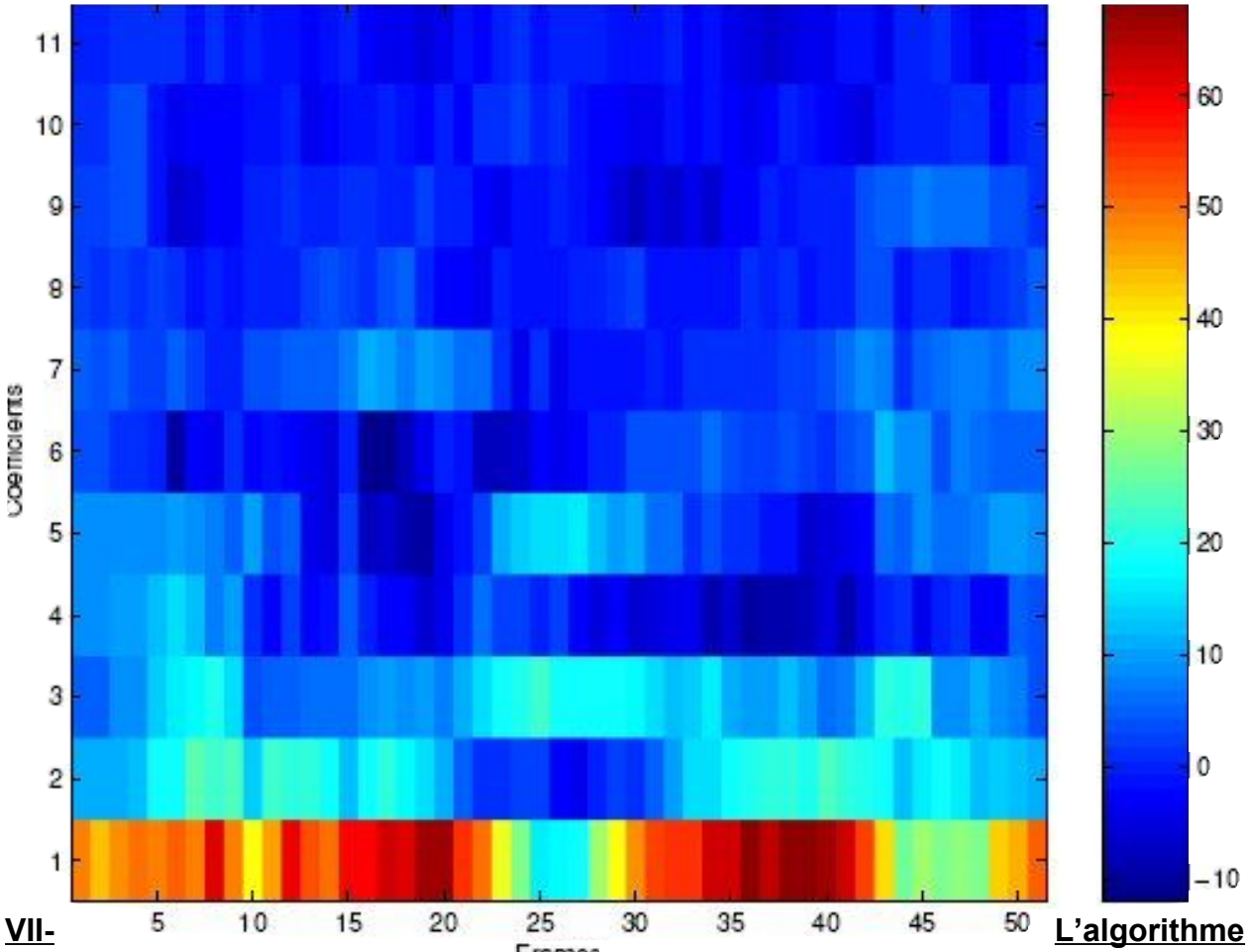
i) Les coefficients Cepstraux.

C'est l'étape finale, on transforme les données dans l'échelle des Mels (fréquencielle donc) vers l'échelle des temps. Le résultat de cette étape sera les MFCC proprement dit. Il suffit d'effectuer l'inverse de la transformée de Fourier. Dans la pratique, on effectue une transformée en Cosinus Discrète inverse (iDCT), ce qui revient au même puisque la transformée en Cosinus inverse donne la partie réelle de la transformée de Fourier ; or ici on a que des réels. Il faut noter que la transformée en sinus donnera la partie imaginaire de la transformée de Fourier.

$$C_{i,n} = \sum_{k=1}^{K} \left(\log_{10} \left(S_{i,k} \right) \right) . \cos \left(n \left(k - \frac{1}{2} \right) \frac{\pi}{K} \right)$$

On exclut le $0^{ème}$ coefficient $\left(C_i, 0 \right)$ car il transporte peu d'informations caractéristiques du signal (il représente la valeur moyenne de l'intensité du signal). On a donc K-1 coefficients.

Il ne reste plus qu'à recoller les frames pour obtenir ce type de Cepstre :



VII- L'algorithme DTW (Dynamic Time Wrapping), comparaison de deux cepstres MFCC.

Pour terminer, reste enfin la dernière étape qui consiste à comparer deux cepstres MFCC, pour cela on utilise l'algorithme DTW (Dynamic time Wrapping). Comme dit précédemment, il est impossible de comparer directement deux spectres (ou cepstres) entre eux, tout simplement parce qu'une même personne ne peut prononcer deux fois le même mot sur la même durée, le même rythme, la même intensité. Il est donc nécessaire de développer une méthode de comparaison cepstre à cepstre par l'algorithme de comparaison dynamique détaillé ci-après.

a) L'algorithme DTW.

Cet algorithme mis en œuvre permet la comparaison dynamique de deux spectres. A cause des variations inévitables entre deux prononciations du même mot, on ne peut pas comparer directement deux signaux. D'abord ils n'auront à coups sûr pas la même durée. C'est pour cela que la comparaison dynamique a été développée (DTW : Dynamic Time Wrapping), aussi appelée normalisation temporelle. On peut distinguer deux sources de variations de l'échelle temporelle : la variation de la vitesse de prononciation et la variation du rythme de prononciation. L'algorithme décrit ici est celui de Vintsyuk proposé en 1968.

Soient *A* et *B* deux images acoustiques (spectres) de longueur *I* et *J* respectivement la « distance » entre l'événement $i \in [1, I]$ de l'événement *A* et l'événement $j \in [1, J]$ de *B* se calcul avec une simple distance euclidienne :

$$d(i, j) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \quad avec \; i = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} et \quad j = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Cela suppose bien sûr que l'on considère la même plage de fréquence pour les deux signaux (entre 0 et *N*). On crée donc un chemin $\{C(K) = (n(k), m(k)), \forall k \in [1, K]\}$. Il est nécessaire que les fonctions $n(k)$ et $m(k)$ soient croissantes et doivent correspondre à certaines contraintes : les seules chemins valides arrivants au point (i, j) sont ceux provenant des points $(i-1, j), (i, j-1)$ et $(i-1, j-1)$. De plus on prend *K* tel que $C(K) = (I, J)$. On pose $C(1) = (1,1)$.

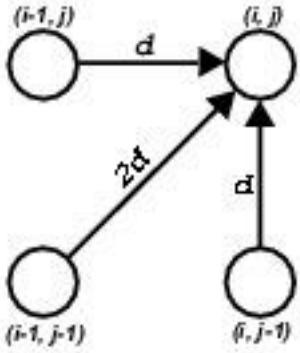
La méthode consiste à choisir le chemin qui passe par les distances $d(i, j)$ les plus petites, de sorte que la distance cumulée le long de ce chemin soit la plus petite possible.

On définit $g(i, j)$ la distance cumulée au point (i, j) comme :

$$g(i, j) = \min \begin{Bmatrix} g(i-1, j) + d(i, j) \\ g(i-1, j-1) + 2.d(i, j) \\ g(i, j-1) + d(i, j) \end{Bmatrix}$$

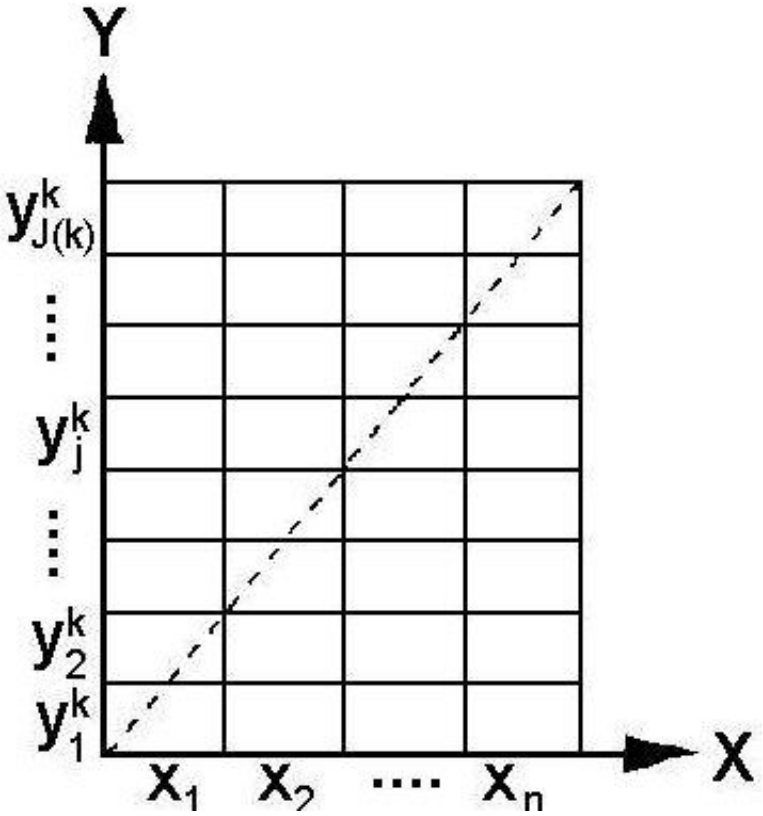
On remplit ensuite la matrice *I* x *J* (le plan du chemin) avec en $i^{ème}$ et $j^{ème}$ colonnes le résultat de $g(i, j)$. Enfin on définit la distance normalisée entre deux prononciations du mot :

$$G = \frac{g(I, J)}{I + J}$$

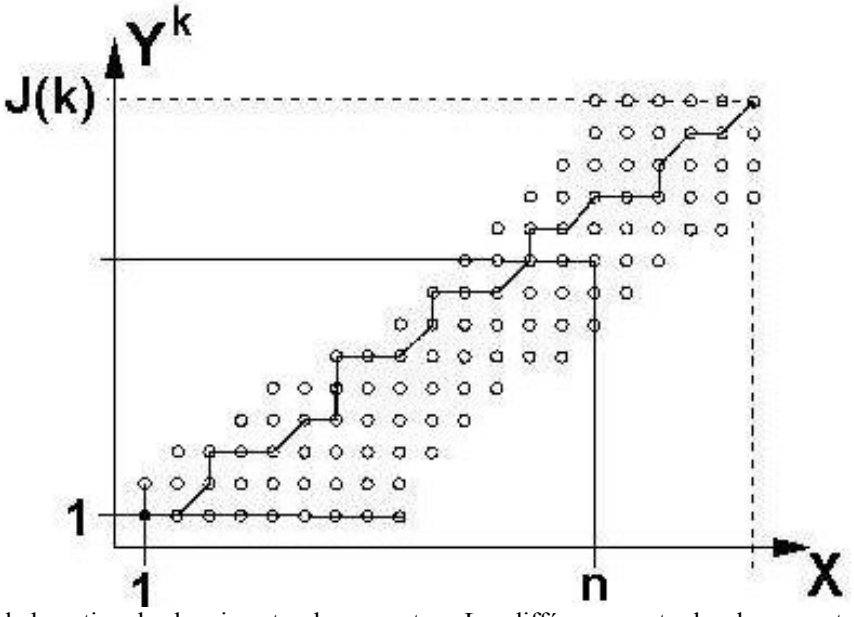


Contrainte de parcours

On obtient une distance entre deux spectres. On effectue ce travail entre le mot à reconnaître et tous les mots du dictionnaire. On prend ensuite le mot du dictionnaire qui a la plus petite distance spectrale avec le mot à reconnaître.



Chemin idéal entre deux spectres



Représentation de la notion de chemin entre deux spectres. Les différences entre les deux spectres « tordent » le chemin idéal (la diagonale).

b) Conclusion.

L'algorithme DTW est un très bon outil capable de comparer deux spectres audio ayant des durées différentes, un débit, une intensité de la voix différente et cela de façon optimale en recherchant le meilleur chemin pour passer d'un spectre à l'autre. Néanmoins d'autres méthodes existent, comme les modèles de Markov cachés (HMM) par exemple bien plus puissant que l'algorithme DTW mais bien plus complexe.

VII- Conclusion sur la méthode de reconnaissance vocale à base de MFCC.

Nous avons donc vu ici une approche globale de la reconnaissance vocale. Au vu des résultats de notre application, on s'aperçoit que cette méthode fonctionne relativement bien. Mais le problème majeur étant que c'est un système mono locuteur et que donc chaque utilisateur doit créer son propre dictionnaire dans la phase d'apprentissage. De plus ce système ne permet pas de reconnaître des phrases mais plutôt des mots isolés. Néanmoins cette méthode peut être parfaitement utilisée dans des appareils d'utilisation courante comme les téléphone portables (utilisation type appel numérotation automatique), les consoles automobiles ou pourquoi pas dans le domaine de la domotique et même pour la commande vocale pour handicapés. D'après notre programme la taille des dictionnaires est très faible, environ 1.5 Ko pour chaque mot et la reconnaissance est très rapide, inférieure à une seconde même avec un dictionnaire de plus de 20 mots.

Il existe des implantations très poussées des MFCC, comme par exemple le système Sphinx. Mais la méthode de comparaison des coefficients MFCC utilisée est statique (modèle de Markov Cachés et aussi approche Neuronaux.

Pour une utilisation plus poussée type dictée vocale ou standard téléphonique automatique nécessitant une reconnaissance de la parole continue, d'autres systèmes doivent être utilisés. Ces systèmes doivent reconnaître instantanément n'importe quelle voix. On utilise donc pour cela une reconnaissance de la parole automatique qui ne considère plus la parole comme une suite de mots isolés. L'ordinateur reconnaît ici les phonèmes qui sont les éléments sonores d'un langage donné, déterminés par les rapports qu'ils entretiennent avec les autres sons de ce langage. La reconnaissance se fait en fonction de contraintes phonétiques, linguistiques : le système élimine les phrases impossible grammaticalement et syntaxiquement grâce à des outils de reconnaissance de formes structurelles (grammaire déterministe). Cette méthode demande donc l'utilisation de traitements de type intelligence artificielle et est difficile à mettre en place, mais les résultats sont très satisfaisants.

IX- Structure des programmes développés dans le cadre de ce PPE.

a) Programme principal.

Le programme principal que nous avons nommé 'Engine' (moteur de reconnaissance vocale) est présenté sous forme de fenêtre MS-DOS. Il nécessite la bibliothèque 'fmod.dll', à copier dans le dossier où se trouve le programme. Le moteur de reconnaissance vocale ne fonctionne qu'avec des fichiers WAV en mono, échantillonnés à 22 050 Hz de qualités 16 Bits. L'utilisation du programme est détaillée ci-dessous avec des exemples :

-p Permet de jouer un fichier audio.

-s Créer un cepstre MFCC qui sera enregistré dans un fichier portant l'extension *.mfc (à définir en ligne de commandes). Ce fichier est optimisé en taille et à son propre format.

-c Permet de comparer deux spectres MFCC entre eux et de renvoyer 'la distance' DTW.

-d Permet de comparer un fichier audio avec un dictionnaire de mots. Le programme renvoie une erreur ou le cas échéant, le mot reconnu.
Un fichier temporaire sous le nom de 'tmp.mfc' sera créé dans le dossier temporaire de Windows.

Exemples :

engine file.wav –p	Joue un fichier audio.
engine file.wav file.mfc –s –p	Créer un cepstre MFCC et joue le fichier audio.
engine file1.mfc file2.mfc –c	Compare deux cepstres MFCC.
engine file.wav dico.txt –d	Compare un fichier audio avec l'ensemble des cepstres MFCC référencés dans un dictionnaire.

La création d'un dictionnaire est assez simple, pour cela vous devez enregistrer un par un chaque mot de votre dictionnaire au format convenu via le microphone de Windows par exemple, c'est-à-dire au format WAV mono, 22 050 Hz en 16 bits. Ensuite, vous devez créer chaque cepstres MFCC de votre dictionnaire en utilisant la fonction de *–s*. Une fois tous les cepstres créés, n'hésitez pas à créer un dossier. Pour terminer il ne vous reste plus qu'à créer un fichier texte comportant l'ensemble des cepstres MFCC formant votre dictionnaire.

Un dictionnaire se présente de cette façon :

```
C:\PPE\dico\avance.mfc
C:\PPE\dico\droite.mfc
C:\PPE\dico\gauche.mfc
...
```

PPE/dico indique le dossier où se trouve les cepstres MFCC (attention n'oubliez pas que les chemins doivent être au format MS-DOS, c'est-à-dire qu'ils ne doivent pas dépasser 8 caractères pour chaque dossiers). Il ne vous restera plus qu'à tester un son avec l'ensemble des mots de votre dictionnaire avec la commande *–d*.

Vous trouverez dans les sources la structure du programme et son déroulement.

b) Programme de dessin de signaux.

Le programme DrawPts a été développé en Visual Basic avec Visual Basic 6.0. Il permet de visualiser n'importe quelle signal audio ou autre. Les fichiers qu'il est capable d'ouvrir portent l'extension *.pts mais en fait il ne s'affiche que de fichiers textes regroupant l'ensemble des coordonnées de points. C'est-à-dire que sur la première ligne on retrouve le point de coordonnées $(1, x_1)$, jusqu'à la ligne n soit la coordonnée du point $n\,(n, X_n)$. DrawPts a été seulement développé dans le but de vérifier le programme principal, ce n'est en aucun cas un programme exploitable.

c) Programme de dessin des spectres.

Le programme DrawSpectre a été développé en Visual Basic avec Visual Basic 6.0. il permet de visualiser un spectre audio fournis directement par le programme 'engine'. Les fichiers qu'il est capable d'ouvrir portent l'extension *.spc mais en fait il ne s'agit que de fichiers textes regroupant les amplitudes des fréquences. DrawSpectre a été seulement développé dans le but de vérifier le programme principal, ce n'est en aucun cas un programme exploitable.

d) Programme de contrôle des ports parallèles.

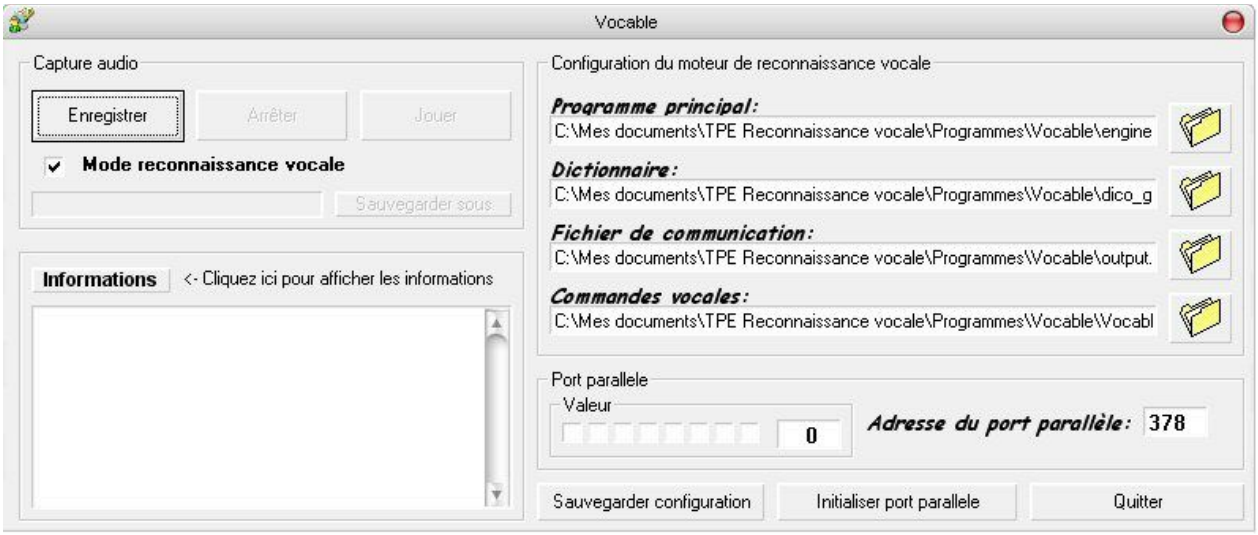
Le programme 'port' a été développé en C/C++ avec Dev-C++, il est présenté sous forme de fenêtre MS-DOS. Il nécessite les fichiers suivants : winio.dll, winio.vxd, winio.sys qui doivent être copiés dans le dossier du programme. Ce petit programme ne permet que de sortir des données du port parallèle.

-a	Choisie l'adresse du registre de données du port parallèle. Ce paramètre est facultatif, par défaut il vaut 0x378. Ce paramètre demande une adresse en hexadécimale de la forme 0h ou 0x.
-v	Ecrit sur 8 bits la valeur désirée sur le registre de données du port parallèle. Ce paramètre est facultatif et par défaut vaut 0. Il initialise donc le port parallèle. La valeur transmise peut être en décimal, binaire (préfixe 0b) ou en hexadécimal (préfixe 0h ou 0x).
-x	x représente le numéro du programme à exécuter. x est compris entre 1 et 4. Ces petits programmes activent une séquence d'allumage des LEDs permettant ainsi de valider ou non le bon fonctionnement de la carte électronique.

`port`	Initialise le port parallèle nommé LPT1.
`port –a 0x278 –v 0b10101010`	Ecrit le nombre 170 sur le port parallèle LPT2 à l'adresse 0x278.
`port –v 0xF0`	Ecrit la valeur 240 sur le port parallèle LPT1 par défaut.
`port –1`	Exécute la séquence d'allumage des LEDs numéro 1..

e) Programme d'interface entre l'homme, le moteur de reconnaissance vocale et le robot.

‘Vocable’ est une interface qui a été développée en Visual Basic avec Visual Basic 6.0. Elle permet de commander le robot de façon simple.



Interface de reconnaissance vocale nommée ‘Vocable’.

Avant d’utiliser cette interface, quelques réglages doivent être effectués.

- ***Programme principal :*** Indique le chemin où se trouve le moteur de reconnaissance vocale (engine.exe).

- ***Dictionnaire :*** Indique le chemin où se trouve le dictionnaire du locuteur qui va commander le robot.

- ***Fichier de communication :*** Ce trouve dans le dossier de engine.exe et porte le nom output.txt, permet une communication entre le moteur de reconnaissance vocale et l’interface.

- ***Commandes vocales :*** Contient les octets a envoyé dans le port parallèle lorsqu’un mot a été reconnu. (Voir l’exemple VocableData.txt dans le dossier du programme).

Vous pouvez sauvegarder les paramètres dans un fichier de configuration en cliquant sur ‘Sauvegarder configuration’, ainsi à la prochaine ouverture vous n’aurez plus besoin de tout reconfigurer.

Si la case ‘Mode reconnaissance vocale’ est décoché, la zone ‘Capture audio’ agit comme le magnétophone de Windows. Dans le cas contraire, votre enregistrement est automatiquement reconnu avec un mot du dictionnaire et une valeur est écrite sur le port parallèle.

Attention : Un bug existe dans l’interface, pour que la reconnaissance vocale automatique fonctionne, vous devez forcer la boîte de dialogue de sélection des fichiers sur le dossier où se trouve ‘engine.exe’. Pour que cela fonctionne, sélectionnez le dossier où se trouve ‘engine.exe’ en modifiant le paramètre ‘Programme principal’.

Il vous êtes possible à tout moment de tester le port parallèle en inscrivant une valeur dans celui-ci par l'intermédiaire des cases à coché se trouve dans la zone 'Port parallèle'.

A noter qu'un fichier enregistré avec 'Sauvegarder sous' sera automatiquement au format WAV, 22 050 Hz, mono en 16 Bits, soit le format de lecture du moteur de reconnaissance vocale.

f) Conclusion

Ces programmes ont été entièrement développés dans le cadre de ce PPE, il ne s'agit en aucun cas de programmes récupérés sur Internet. Ces programmes peuvent être dangereux pour le matériel si ils sont mal utilisés. C'est le cas pour 'Port' ou encore 'Vocable', si mal utilisés, peuvent détruire le port parallèle. Nous ne sommes en aucun cas responsable des dommages dû à une mauvaise manipulation.

X- Sources.

http://www.bruno.mathieu.freesurf.fr/These/node7.html
http://philduweb.free.fr/contributions/parole/analyse.htm
http://www.cem2.univ-montp2.fr/cours/ProjetsIUP2/C5/rapportfinal/rapport.htm
http://www.ensicaen.ismra.fr/~furon/_traitementsignal/_cours_tns/_fenetrage/005.htm
http://r.battault.free.fr/probatoire/probatoire.html
http://liocity.free.fr/charger_delphi/tutorial/tutoriel33_fft.htm
http://www.vieartificielle.com/article/index.php?id=191
http://perso.wanadoo.fr/philippe.baucour/pratiquer/fft/fft1.html
http://tcts.fpms.ac.be/cours/1005-08/speech/projects/2001/delfabro_henry_poitoux/
http://www.xeberon.net/index.php