

Rapport de ModEx  
PHY 434  
Reconnaissance vocale

Stéphane BELARDI  
Olivier ROBIN

13 avril 2005

# Table des matières

<b>1</b>	<b>Présentation du problème posé</b>	<b>2</b>
<b>2</b>	<b>Caractéristiques communes des sons étudiés</b>	<b>3</b>
<b>3</b>	<b>Nos premières approches</b>	<b>5</b>
3.1	Méthode des moindres carrés sur le spectre . . . . .	5
3.2	Cepstre . . . . .	6
<b>4</b>	<b>Approche finale : fonctionnement</b>	<b>8</b>
4.1	Fonctionnement de la voix (filtres) . . . . .	8
4.2	Courbe LPC . . . . .	9
4.2.1	Théorie . . . . .	9
4.2.2	Algorithme . . . . .	11
4.2.3	Problèmes rencontrés . . . . .	11
4.3	Méthodes des pôles (LPC) . . . . .	15
4.3.1	Fonctionnement . . . . .	15
4.3.2	Ce qui n'a pas marché . . . . .	16
<b>5</b>	<b>Applications : le mot de passe vocal et la calculatrice vocale</b>	<b>18</b>
5.1	Mot de passe vocal . . . . .	18
5.2	La calculatrice vocale . . . . .	18
5.2.1	Le principe . . . . .	18
5.2.2	L'implémentation et les résultats . . . . .	19
5.2.3	les problèmes spécifiques à la calculatrice . . . . .	19
<b>6</b>	<b>Conclusions et remarques</b>	<b>21</b>
6.1	Ce qu'on aurait pu faire de plus . . . . .	21
6.2	Le ModEx . . . . .	21
6.3	Conclusion . . . . .	22

# Chapitre 1

## Présentation du problème posé

Les films futuristes, ou les films dont la composante technologique est accentuée, ne peuvent se passer de présenter à un moment ou à un autre une reconnaissance de personne par la voix. Ces reconnaissances semblent alors parfaites, sans aucune erreur, totalement automatiques, et relativement rapides. La société *Domain Dynamics*, spécialisée dans le traitement des signaux et dans les technologies de reconnaissance vocale a d'ailleurs inventé en 2000 une carte à puce à mot de passe vocal (celle-ci ne fonctionne qu'avec une puce Java standard 8 bits, 8ko, un algorithme de 3ko et la reconnaissance prend quelques millisecondes ; cependant, l'exactitude ne nous est pas connue).

Nous avons donc tenté, avec certes des moyens limités par rapport aux deux exemples précédents, d'écrire un algorithme écrit en langage *MatLab*<sup>®</sup> qui, à partir d'un bref apprentissage et d'un échantillon de voix, doit :

- reconnaître la personne qui avait prononcé l'échantillon de la voix avec une exactitude supérieure à 70%
- vérifier si le mot prononcé est le mot de passe de la personne avec une exactitude supérieure à 60%
- exécuter ces deux étapes assez rapidement (moins de 30 secondes sur un PC classique)
- exécuter ces deux étapes automatiquement (la participation de l'utilisateur doit être limitée)



FIG. 1.1 – La reconnaissance vocale au cinéma

## Chapitre 2

# Caractéristiques communes des sons étudiés

Nous avons dès à présent défini “quelques standards” pour tous nos sons (qu’ils soient sons du dictionnaire ou sons à tester) :

- Nous avons fixé la fréquence d’échantillonnage à  $44100\text{Hz}$ . Mais notre code est censé s’adapter aux fréquences d’échantillonnage différentes ( $22050\text{ Hz}$ ,  $11025\text{ Hz}$ , ...). Le son doit être également monophonique (et le code ne supporte pas un son stéréophonique).
- Nous avons régulièrement travaillé sur des intervalles d’analyse (voir figure 2.1) de 1024 échantillons (soit environ  $23\text{ms}$ ). Cela correspond à des durées sur lesquelles on peut considérer le signal de la voix comme stationnaire, en tout cas, si c’est une voyelle qui est prononcée. En effet, la prononciation d’une consonne est un phénomène transitoire, qui donne de mauvais résultats lors du calcul des coefficients LPC.
- Enfin, un fenêtrage est nécessaire lors de calculs de spectres ou de LPC. L’ajout de ce fenêtrage lors des calculs a nettement amélioré les résultats. En revanche, le changement de fenêtre n’apporte que peu (voire pas) de changements. Nous avons choisi finalement la fenêtre de Blackmann-Harris, car c’est celle qui est utilisée par défaut dans le logiciel *Adobe® Audition®*.

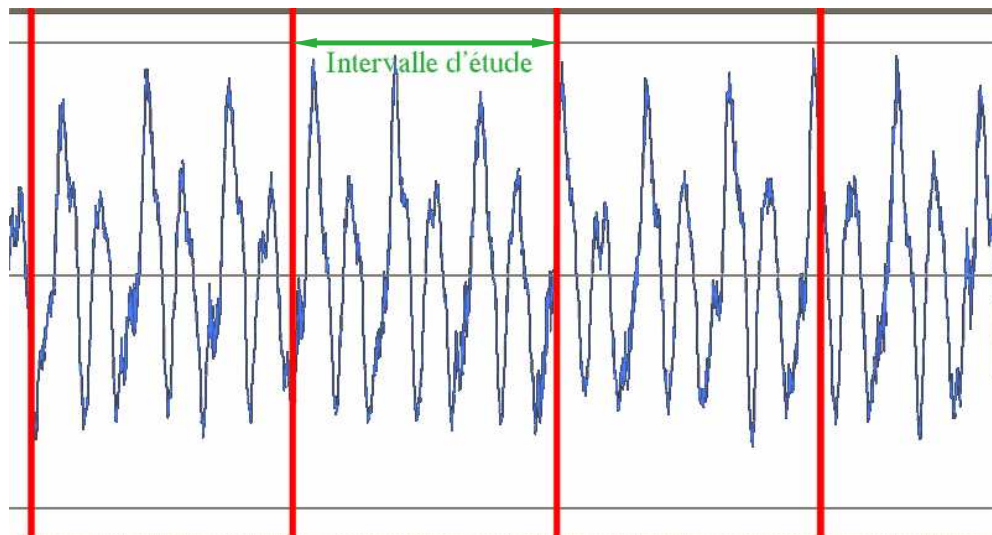


FIG. 2.1 – Intervalles d’étude

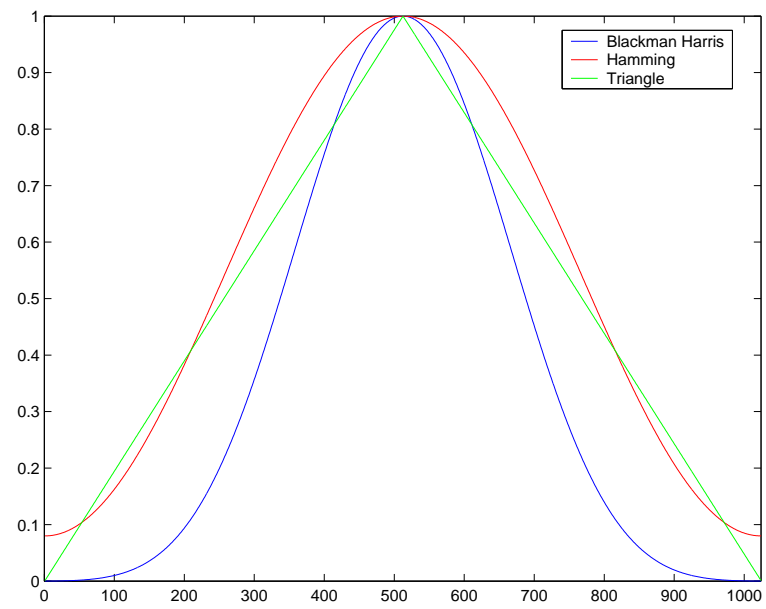


FIG. 2.2 – Exemples de fenêtres

## Chapitre 3

# Nos premières approches

### 3.1 Méthode des moindres carrés sur le spectre

La première méthode testée, que l'on pourrait qualifier de “naïve”, a été la méthode des moindres carrés sur le spectre. Une méthode encore plus naïve aurait été de comparer les signaux temporels, mais cette méthode aurait donné des résultats moindres que ceux obtenus grâce au spectre fréquentiel.

**Principe** Le spectre est une représentation fréquentielle du son : c'est la fonction qui à toute fréquence fait correspondre son amplitude dans l'intervalle d'analyse<sup>1</sup>.

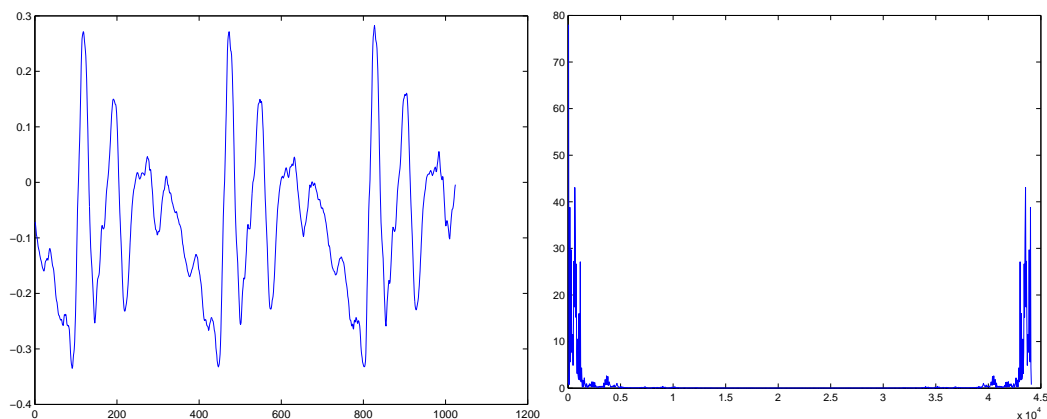


FIG. 3.1 – Signal temporel et spectre d'un échantillon de 1024 points prononcé par une personne A

Nous avons calculé ensuite le spectre de chacun de ces secteurs, puis fait la somme des carrés des différences entre un spectre “dictionnaire” et un spectre de l'échantillon ( $\sum_f |s_1(f) - s_2(f)|^2$ , où  $s_1$  et  $s_2$  sont les 2 spectres).

**Problèmes** En effet, la fonction carré est celle qui donnait le meilleur résultat, qui n'était que d'environ 25%.

Un simple raisonnement permet de comprendre pourquoi : un simple décalage dans la fréquence (qui correspond à une prononciation à peine plus aiguë ou plus grave) rend cette méthode caduque.

De plus, nos spectres étaient également très bruités et comportaient de fortes variations, ce qui rendait les différences quasiment aléatoires.

Enfin, cette méthode considère toutes les fréquences à un niveau égal, alors que la voix ne produit que peu de fréquences supérieures à  $8kHz$ . Nous avons testé d'autres fonctions qui se comportent différemment suivant la fréquence et suivant la différence. Voici quelques exemples :

---

<sup>1</sup> Analyse de Fourier

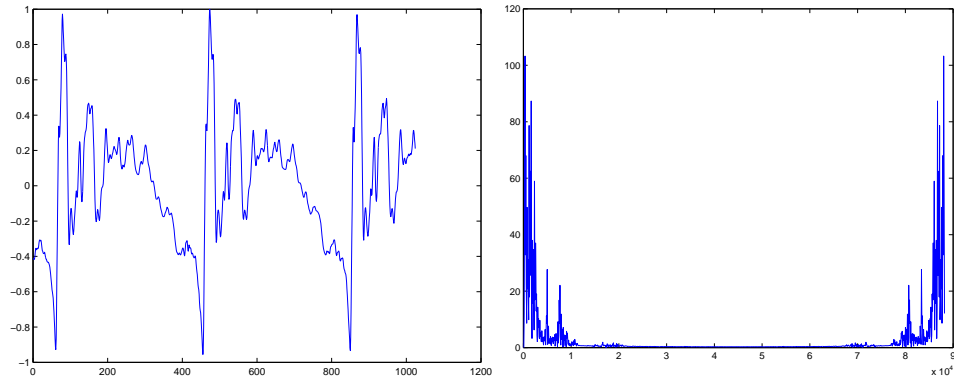


FIG. 3.2 – Signal temporel et spectre d'un échantillon de 1024 points prononcé par la même personne A

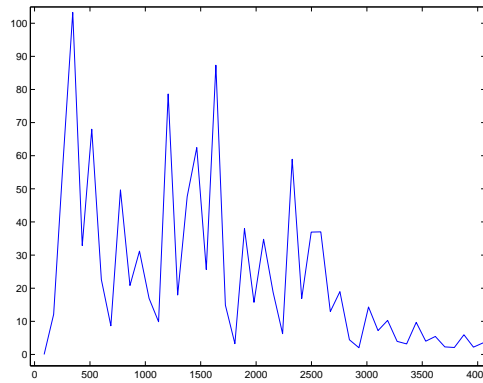


FIG. 3.3 – Exemple de spectre bruité

$$\begin{aligned} & \sum_f \log(|s_1(f) - s_2(f)|) \\ & \sum_f \frac{|s_1(f) - s_2(f)|^2}{f} \\ & \sum_f \frac{|s_1(f) - s_2(f)|^2}{\log(f)} \end{aligned}$$

Mais celles-ci ne donnent pas de résultats convaincants.

## 3.2 Cepstre

**Principe** En se basant sur le fonctionnement de la voix (cf 4.1), on peut supposer qu'un signal prononcé est de la forme  $excitation(t) \star filtre(t)$ , où l'excitation est créée par les cordes vocales et le filtre modulé par les variations de positionnement de la langue et de la gorge. Pour déconvoluer ce produit, il existe la méthode dite du cepstre. En effet, celle-ci est basée sur l'opérateur logarithme qui transforme un produit en addition, ce qui permet alors de séparer la source du conduit vocal.

$$Cepstre(k) = TF^{-1}(\log |TF_{signal}(f)|^2)$$

où l'on a noté  $TF$  la fonction Transformée de Fourier. Les arguments de la fonction cepstre sont appelés "quéfrences", homogènes à un temps.

**Problèmes** On obtient donc une somme de l'excitation et du filtre. Il existe plusieurs méthodes pour les séparer, la plus courante étant celle du liftrage. Cette méthode consiste à supprimer les

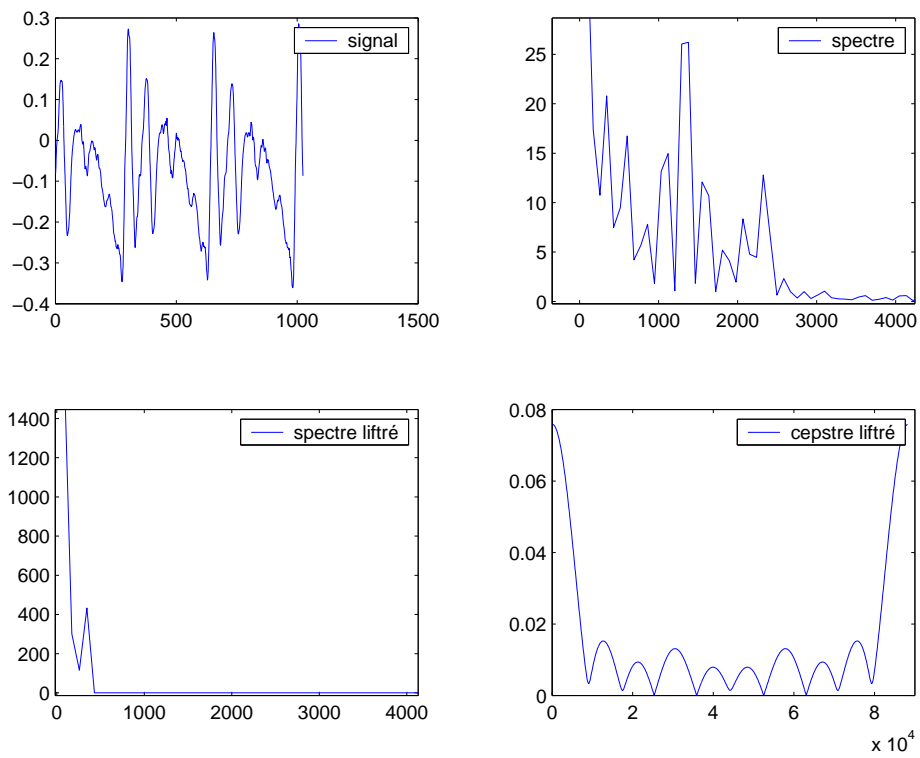


FIG. 3.4 – Étapes de calcul pour l'obtention du cepstre

quéfrenes à partir d'un certain seuil (entre 250 et 400 Hz pour de bons résultats). Mais nous n'avons jamais eu des courbes satisfaisantes.



## Chapitre 4

# Approche finale : fonctionnement

Pour obtenir de meilleurs résultats, il est nécessaire de s'intéresser au fonctionnement et à la nature même de la voix.

### 4.1 Fonctionnement de la voix (filtres)

L'appareil phonatoire humain peut être représenté par un système linéaire arbitrairement complexe, excité par un signal déterminé. Il suffit alors de déterminer les coefficients du système linéaire et le signal d'excitation le plus approprié pour parvenir à une approximation valable du signal original, qui est un signal de parole. La fonction LPC de MatLab permet de calculer ces coefficients. S'il est raisonnable de modéliser ainsi la voix humaine, il suffit alors en principe de ne transmettre que la série de coefficients que l'on a calculés et le type du signal d'excitation pour transmettre entièrement un signal à partir duquel on peut reconstituer exactement la voix initiale. L'appareil

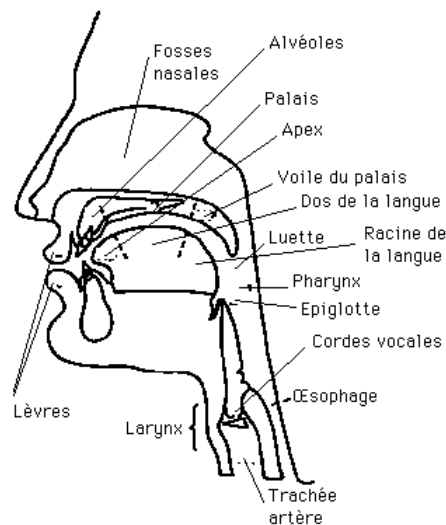


FIG. 4.1 – L'appareil phonatoire de l'homme

respiratoire fournit l'énergie nécessaire lorsque l'air est expiré par la trachée-artère. Au sommet de celle-ci se trouve le larynx où la pression de l'air est modulée (voir figure 4.2) avant que celui-ci ne soit expulsé par le conduit vocal qui s'étend du pharynx jusqu'aux lèvres. Le larynx est un ensemble de muscles et de cartilages mobiles qui entourent une cavité située à la partie supérieure de la trachée. Les cordes vocales sont en fait deux lèvres symétriques placées en travers du larynx ; ces lèvres peuvent fermer complètement le larynx et, en s'écartant, déterminer une ouverture triangulaire appelée glotte. Le larynx, le pharynx et la bouche forment trois cavités qui fonctionnent comme des filtres lors du passage du train d'onde exciteur.

Les sons voisés résultent au contraire d'une vibration périodique des cordes vocales ; des impulsions périodiques de pression sont ainsi appliquées au conduit vocal qui est un ensemble de cavités situées entre la glotte et les lèvres. Il peut être considéré comme une succession de tubes ou cavités

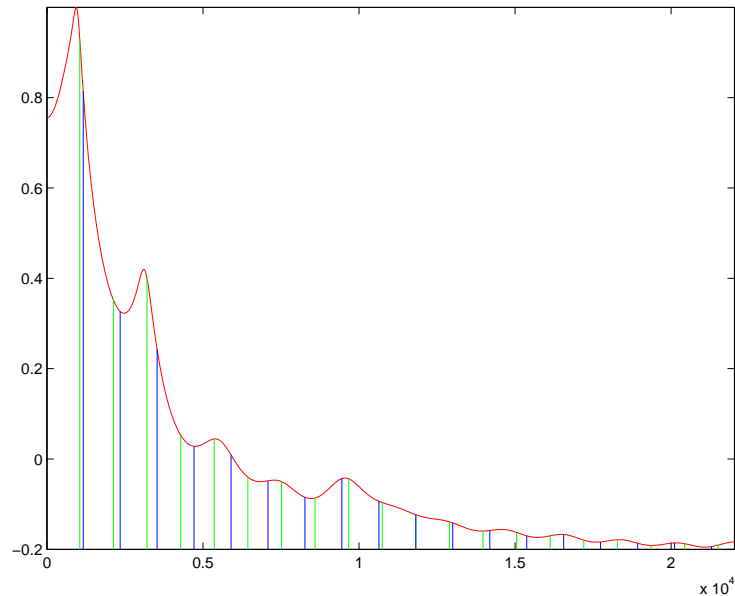


FIG. 4.2 – Un filtre LPC et la réponse à une excitation

acoustiques de sections diverses. Les sons voisés résultent donc de l'excitation du conduit vocal par des impulsions périodiques de pression liées aux oscillations des cordes vocales.

Dans la suite, nous nous sommes intéressés essentiellement aux sons voisés, qui sont beaucoup plus faciles à traiter car de durée plus longue.

L'oreille fonctionne comme un décodeur de ces filtres. Elle tente de déterminer les premiers pics, appelés formants, de leur réponse fréquentielle afin de reconnaître la voyelle. C'est exactement ce que nous allons essayer de reproduire informatiquement.

## 4.2 Courbe LPC

Afin de prendre en compte le processus de formation de la voix, on décide de regarder les coefficients LPC. Ceux-ci correspondent au filtre formé par les cavités par lesquelles passe le train d'impulsions initiales.

Matlab permet d'obtenir ces coefficients directement. Nous pouvons ensuite regarder la réponse fréquentielle d'un tel filtre. On s'aperçoit rapidement que les coefficients sont caractéristiques par deux aspects :

- Par la fréquence des pôles qui détermine les fréquences les moins atténuées par ce filtre
- Par les gains relatifs de ces différents pôles

Par exemple, on voit sur la figure 4.3 les réponses fréquentielles des filtres des différentes voyelles.

On voit sur ces graphiques les différences entre les fréquences des pôles<sup>1</sup> des différents filtres. Ce sont ces pôles qui vont nous permettre de reconnaître la lettre énoncée.

Par exemple, sur le tableau de la figure 4.4 on peut voir les différentes fréquences des premiers pôles des différentes voyelles.

### 4.2.1 Théorie

Notre question théorique porte sur ce point : quelle distance permet de différencier les différents formants codant les voyelles ?

Nous sommes donc partis sur plusieurs pistes.

#### L'approche naïve

La première approche lorsqu'on doit comparer deux fonctions est la méthode des moindres carrés. Nous sommes donc partis de là. Le son à tester, une fois décomposé et analysé est comparé,

<sup>1</sup>Un pôle du filtre correspond à un pic sur la fonction

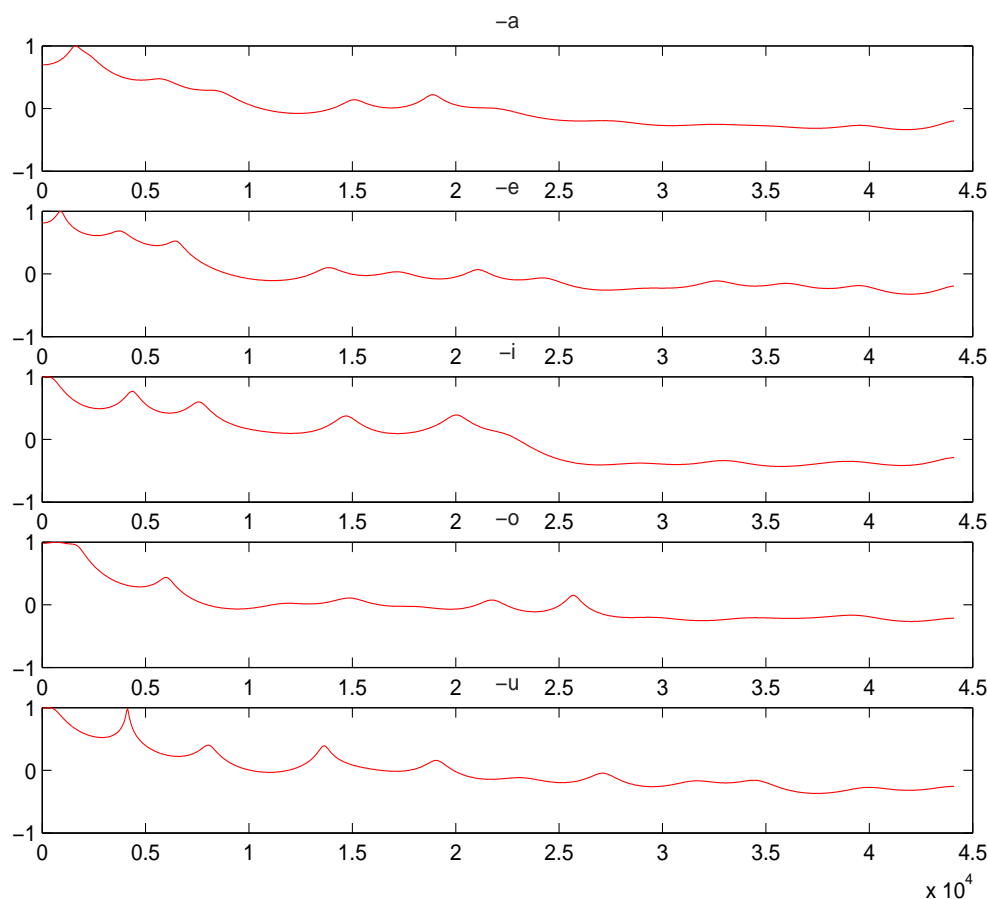


FIG. 4.3 – Les différentes réponses pour les différentes voyelles

A	→	1590 Hz
E	→	905 Hz
I	→	215 Hz
O	→	690 Hz
U	→	345 Hz

FIG. 4.4 – Les premiers pics des différentes voyelles

fréquence par fréquence aux sons du dictionnaire et la somme des carrés de la différence nous donne une distance globale. Le son de référence le plus proche nous donne la voyelle prononcée.

Cette approche s'est révélée, comme on pouvait s'y attendre infructueuse. Dire qu'on avait 20% de réussite montre à quel point ce procédé est aléatoire. Nous avons donc essayé de comprendre pourquoi cette distance ne semblait pas marcher.

### Améliorations

Une analyse de plusieurs courbes nous a amenés à constater que les réponses des filtres n'étaient pas exactement les mêmes. En fait, cela dépendait des gammes de fréquence. Pour les basses fréquences, les courbes sont pratiquement identiques, alors que pour les hautes fréquences, leur comportement semble aléatoire. Cela semble provenir de l'origine physique de ces pôles. Les premiers sont les pôles provenant des cavités de la gorge, dont la géométrie varie peu. Les derniers proviennent de la bouche qui, elle, est à géométrie hautement variable. Pour augmenter l'importance des basses fréquences, nous avons multiplié la distance par une fonction en  $\frac{1}{N}$ . Les résultats ont été immédiatement meilleurs, passant de 20% à environ 80%.

Des affinements ultérieurs nous ont amenés à modifier la fonction en  $\frac{1}{N}$  et à donner un peu moins d'importance au début de la courbe. Mais le principe était trouvé.

Nous avons ensuite essayé deux méthodes de distance :

- La première consiste à effectuer l’opération sur tous les groupes de 1024 échantillons, puis à en faire la moyenne. Cela atténue considérablement les problèmes dus aux attaques des sons, ou les légères différences qui peuvent résulter d’un bruit de fond. Cependant, cela diminue également les caractéristiques des formants, et les filtres en deviennent moins caractéristiques. Cependant cela fournit pour la distance un indicateur de fiabilité du points. En divisant la distance par l’écart-type des signaux de référence, on prend en compte une telle fiabilité.
- La seconde est d’effectuer l’analyse sur chaque groupe de 1024 échantillons et de stocker la lettre obtenue pour chacun d’eux. Cela permet de garder les caractéristiques de chaque son intactes, mais augmente également les déviations à la norme qui rendent l’identification difficile. Cela produit également des effets incontrôlables lors des attaques des sons, où les consonnes rendent les résultats difficiles à interpréter. En revanche, cette méthode facilite grandement par la suite la séparation des syllabes. On peut par exemple qualifier une syllabe par sa stabilité sur  $300ms$  (10 à 15 groupes d’échantillons).

### La méthode des pôles

Une approche suivante nous a amenés à considérer les pôles seuls. Nous avons testé une distance qui consiste à prendre les pôles de la fonction de transfert du filtre et à les comparer avec ceux du dictionnaire. Nous avons aussi prévu de classer ces pôles suivant leur partie réelle, qui est l’équivalent de la fréquence dans ce cas.

Nous n’avons malheureusement pas réussi à aboutir de façon convaincante. Outre les problèmes de séparation des pôles qui seront exposés au paragraphe 4.3, les résultats ne nous ont pas satisfaits. Même sur le pôle de plus grande partie réelle, dont l’affiche aurait dû être fixe, il existait des différences. Cela aurait pu être amélioré par l’augmentation de l’ordre du LPC, mais nous n’aurions alors plus pu séparer les différents pôles.

### 4.2.2 Algorithme

Comme pour les premiers modèles, nous avons décidé de commencer par l’algorithme le plus simple possible. Nous avons donc travaillé sur la réponse fréquentielle. Le son est découpé en intervalles d’analyse de 1024 échantillons, soit  $23ms$  pour la majorité de nos enregistrements (44,  $1kHz$ ). Nous effectuons ensuite le traitement LPC sur chacun de ces intervalles d’analyse. Pour des raisons de temps de calcul et de précision de la solution, nous avons choisi de calculer les coefficients LPC à l’ordre 25.

#### Le cœur du programme, l’algorithme LPC

```
for i = 1:size(n,2)
    coefficientsLPC(i,:) = LPC(n(:,i),25) ;
    [ReponseFrequentielle(:,i), f(:,i)] = freqz(1, coefficientsLPC(i,:), N, Fe) ;
end Gain = log10(abs(Gain)) ;
```

### 4.2.3 Problèmes rencontrés

#### Séparation des périodes de voix et de non voix

L’un des plus gros problèmes rencontré lors de ce ModEx a été de séparer dans le signal ce qui relève de la voix de ce qui relève du bruit.

N’étant pas toujours dans des conditions optimales pour les enregistrements, le bruit a souvent parasité les données, et affecté les résultats à un degré qu’il est difficile d’évaluer.

A ce stade, deux solutions se présentaient à nous :

- Considérer que tous les fichiers ne contenaient que du son, et les couper en conséquence. Cela est un raccourci certain de programmation mais posera par la suite des problèmes, en particulier lors de la reconnaissance de mots. De plus, les blancs sont souvent le signe de changements de voyelles, comme nous le verrons au paragraphe 4.2.3.
- S’attaquer au problème et tenter de trouver différents moyens de discriminer les sons.

L’enjeu de cette séparation, et le but premier de ce ModEx nous a rapidement fait faire le second choix. Nous avons essayé différentes méthodes pour séparer le son du “non-son”.

**Première méthode : l'amplitude** Cette méthode est très simple : on repère l'amplitude maximale du signal, et on considère comme son toute fenêtre de 1024 échantillons dont l'amplitude est supérieure au maximum - 30dB.

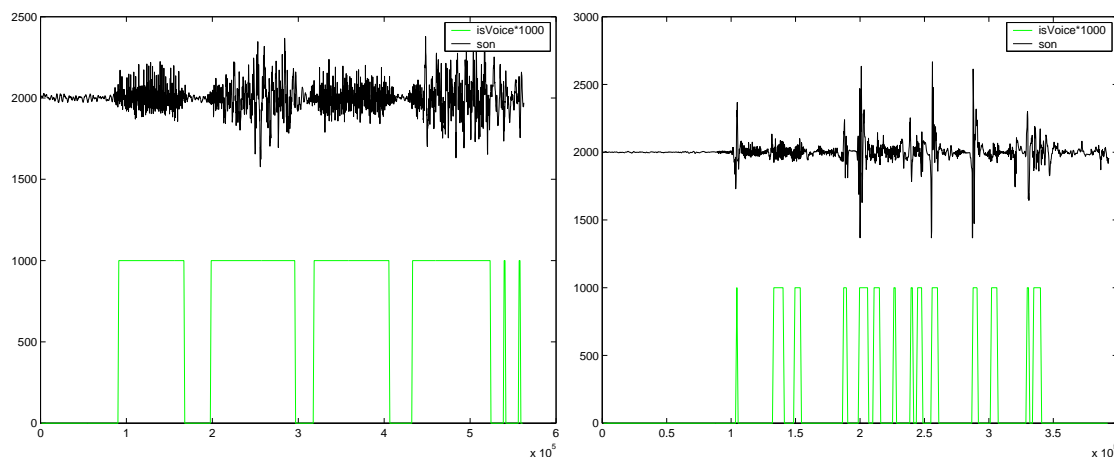


FIG. 4.5 – Un bon et un mauvais découpage par la séparation en amplitude

- Avantages
  - L'algorithme est très simple
  - Sur un son standard, où le locuteur parle d'une voix d'un volume constant, et où aucun bruit parasite ne gêne l'enregistrement, il s'agit de la méthode qui rend le meilleur résultat. C'est d'ailleurs pour cette raison que l'on choisira ce procédé pour l'implémentation de la calculatrice (cf paragraphe 5.2), programme pour lequel les mots enregistrés sont très courts, et où on a donc besoin d'une grande précision de séparation.
- Inconvénients
  - Cette méthode ne fait en fait pas la différence entre le bruit et la voix. Ainsi, un bruit blanc suffit à la tromper.
  - De plus, la voix est très rarement d'un volume constant. En particulier, lors d'une attaque de consonne (surtout pour les p, les t...), le volume augmente brusquement et corrompt le résultat (voir figure 4.5)

**Deuxième méthode : la fréquence fondamentale** L'idée est ici qu'un son voisé possède un harmonique fondamental bien défini de basse fréquence, alors que le bruit en a un chaotique et de haute fréquence. On calcule donc pour chaque fenêtre de 1024 échantillons la fréquence fondamentale par autocorrélation. Est déclaré son tout signal de fréquence fondamentale entre 100 et 1000 Hz.

- Avantages
  - Encore simple à programmer
  - Reconnaît un bruit blanc comme un bruit
- Inconvénients
  - La fréquence fondamentale correspond au premier pic de l'autocorrélation (voir 4.2.3). Le déterminer n'est pas chose aisée. De plus, on tombe assez souvent sur un pic de fréquence double ou triple qui fausse le calcul
  - Il arrive que du bruit ait une fréquence fondamentale entre 100 et 1000Hz. Il est alors reconnu comme mot.

Nous avons finalement utilisé un mélange de ces deux premières méthodes :

- Un premier filtrage est effectué sur l'amplitude pour enlever les zones où le son est trop faible
- On effectue ensuite un filtrage à la fois sur la fréquence fondamentale en prenant soin de considérer les fréquences multiples comme valides.
- On ajoute un filtre sur la variation de la fréquence fondamentale

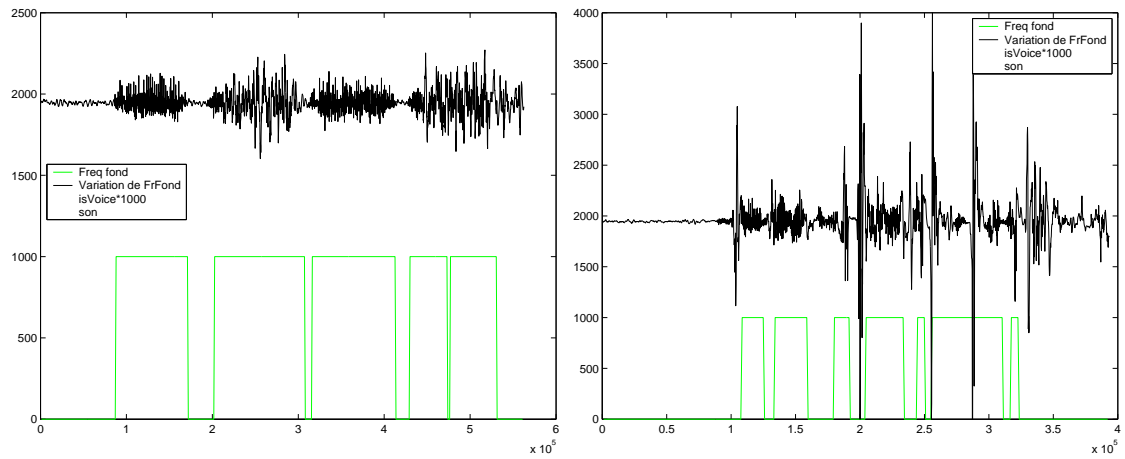


FIG. 4.6 – Un bon et un mauvais découpage par la séparation en fréquence

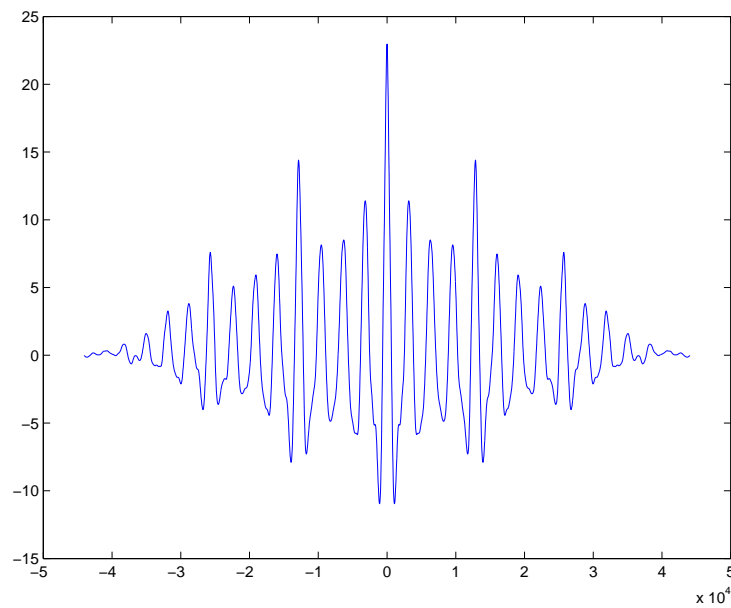


FIG. 4.7 – Graphique d'autocorrélation

### La fréquence fondamentale

La détermination de la fréquence fondamentale se fait par autocorrélation. Le principe est simple :

On calcule :

$$g(\tau) = \int_{-\infty}^{\infty} f(x) \cdot f(x - \tau)$$

Lorsque  $\tau$  est égal à une période de  $f$ , alors  $g(\tau) = \|f\|^2$  et est alors maximum. On cherche donc la fréquence du premier pic (voir figure 4.7).

Le premier pic<sup>2</sup> est détecté par un algorithme très simple :

```

frequence=0; curseur=1025;remonte=0;
while frequence==0
    if f(curseur+10,i)>f(curseur,i)
        remonte = 1;
    end
    if f(curseur+10,i)<f(curseur,i) & remonte==1
        frequence = curseur+5;
    end
end

```

<sup>2</sup>Qui n'est pas forcément celui de plus grande amplitude

```

end
curseur=curseur+1;
end
frqFond(end+1) = frequence;

```

Malheureusement, cette méthode tombe parfois sur un pic secondaire de fréquence plus élevée. Elle n'est donc pas infallible.

## Séparation des syllabes

Un autre problème a été la séparation des syllabes. Apparue plus tard dans le ModEx, au moment où l'on a commencé la reconnaissance de mots de passe, cette séparation des syllabes a été plus difficile que prévu.

Deux cas se sont présentés (voir figure 4.8) :

- Dans le premier cas, heureusement le plus fréquent, les différentes syllabes sont séparées par un court espace qui permet de bien les différencier. C'est notamment toujours le cas lorsque deux voyelles consécutives sont les mêmes. Afin de bien reconnaître ces coupures, nous avons fait appel à une méthode plus simple pour reconnaître la voix, basée uniquement sur l'amplitude.
- Dans d'autres cas, il n'y a pas de blanc entre les syllabes. Il faut alors trouver une autre méthode. La méthode retenue est la suivante :

1. On enregistre le mot de référence
2. On fait reconnaître chaque groupe de 1024 échantillons
3. On note manuellement la taille des différentes syllabes
4. Le mot est reconnu si la proportion des syllabes est la même

Cela permet de mettre un filtre supplémentaire sur le mot : une même personne mettra toujours les mêmes importances relatives pour les syllabes d'un même mot.

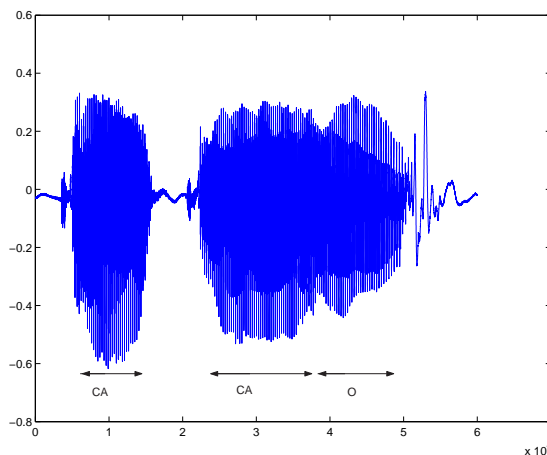


FIG. 4.8 – Un mot typique : cacao

## Reconnaissance des lettres et du locuteur

Nous avons choisi d'axer la reconnaissance des lettres et du locuteur uniquement sur les filtres LPC. L'analyse du train d'onde excitateur, si elle avait pu amener d'autres éléments, s'est révélée difficile par l'inconstance de celui-ci pour une même personne ou une même lettre.

La reconnaissance est donc effectuée en six étapes (voir paragraphe 5.1).

Si cette méthode marche théoriquement à 100%, les pôles étant à la fois caractéristiques de la personne et du phonème prononcé, il n'en est pas de même en pratique. Les perturbations sonores, les changement de voix de chaque locuteur, la diversité des micros (pas moins de 5 microphones différents ont été utilisés lors des différentes phases du ModEx, la reconnaissance n'étant optimale que pour la comparaison de sons enregistrés avec le même micro) font que les pourcentages de réussites sont assez éloignés de l'unité.

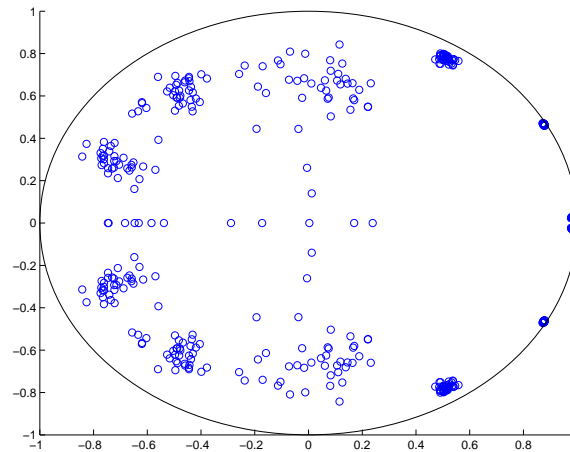


FIG. 4.9 – Les pôles d'un "O"

De plus, l'algorithme LPC est au départ optimisé pour un bruit. Il peut être assez performant pour des mots, mais son efficacité baisse considérablement lorsqu'il s'agit de reconnaître des voyelles nasales ("an", "in", ...). De plus, notre algorithme confond souvent certaines lettres, comme "e" et "o"; "ou" et "u", ...

### Ressemblance de locuteurs

Un des problèmes pratiques qui nous a le plus retardés a été la ressemblance des voix de locuteurs. Les premiers essais ont été faits sur des échantillons de voix de Stéphane BELARDI et Olivier ROBIN, et nous avons obtenu des résultats médiocres. Nous avons donc considéré que les programmes étaient inefficaces. Par la suite, nous nous sommes aperçus que nos deux voix étaient très proches du point de vue de la distance considérée, et que les distances étaient de 10 à 100 fois plus faibles qu'avec toute autre voix.

Si ce problème peut paraître anecdotique, il est à la source de changements inutiles, mais aussi d'essais de distances différentes pour résoudre ce problème. Les améliorations ont été limitées, mais l'étude a servi par la suite pour séparer des sons dans le cas de locuteurs plus nombreux.

## 4.3 Méthodes des pôles (LPC)

### 4.3.1 Fonctionnement

Comme nous l'avons vu, la distance entre les différents enregistrements se calcule à partir des pôles de la réponse au filtre LPC. Pourquoi alors ne pas faire les calculs uniquement à partir de ces pôles? Cela présente un double avantage :

- On se limite dans les informations des filtres LPC à ce qui est essentiel, en éliminant au passage ce qui pourrait avoir perturbé la mesure
- Le stockage est plus aisé car plutôt que de stocker les 1024 points de la réponse, il suffit de stocker les 13 pôles du filtre<sup>3</sup>.

Cependant, il est utile de noter que ce faisant, on masque également de l'information pouvant avoir son utilité. En particulier, l'atténuation relative des pôles (surtout pour ceux éloignés du cercle unité) nous est apparue plus difficile à évaluer.

L'étude par les pôles se fait comme suit :

1. On calcule le filtre LPC du son considéré
2. Grâce à la commande de MatLab, on calcule les pôles complexes du filtre (voir figure 4.9)
3. On classe les pôles par partie réelle décroissante
4. On fait la somme des distances des pôles de l'échantillon testé au barycentre des pôles des échantillons de référence

<sup>3</sup>Comme nous le verrons plus tard, les pôles sont conjugués. Pour un filtre d'ordre 25, il y a donc 2\*12 pôles conjugués et un pôle réel



Afin de tenir compte comme dans la méthode précédente de l'importance inégale des différents pôles, on divise les distances par :

- la partie réelle du pôle
- un numéro d'ordre du pôle

### 4.3.2 Ce qui n'a pas marché

Le point le plus problématique de cette méthode a été l'étape numéro 3. Aucun algorithme n'a réussi à regrouper efficacement les pôles des différents groupes d'un même enregistrement. Pour bien comprendre le problème, il faut s'intéresser à la structure des pôles. Les pôles sont soit :

- des complexes conjugués de module environ 1 correspondant aux caractéristiques de la voix (plus le module est proche de 1, plus l'amplification par le filtre est importante)
- des nombres presque réels<sup>4</sup> compris entre  $-1$  et  $1$

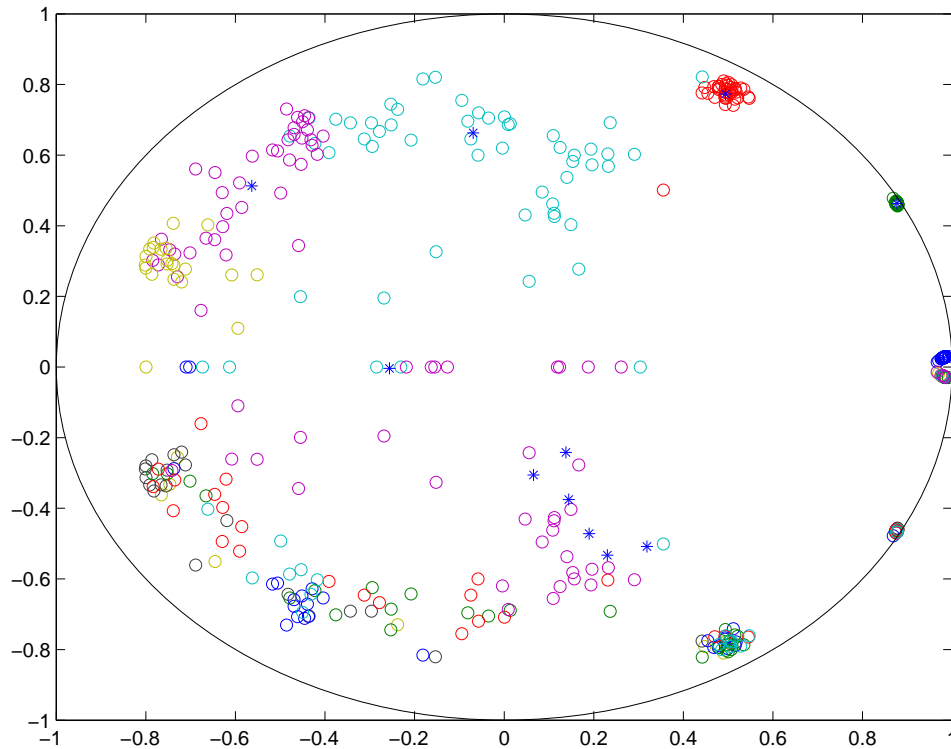


FIG. 4.10 – Les pôles d'un "O"

Cela a été causé par trois facteurs principaux que nous avons identifiés :

- L'ordre du traitement LPC. Cette méthode exige un compromis. On peut soit :
  - augmenter l'ordre du LPC, ce qui signifie des pôles d'une meilleure précision, des écarts types plus faibles, mais aussi plus de pôles, et donc des groupes qui peuvent se chevaucher
  - diminuer l'ordre, on a alors moins de groupes, plus distincts, donc mieux séparables, mais aussi plus diffus.

Nous n'avons pu trouver de valeur donnant des pôles suffisamment précis et séparables pour pouvoir servir dans notre étude

- Le nombre de pôles réels n'est pas constant tout au long de l'enregistrement. Or, nous ne pouvons pas utiliser ces pôles lors de notre traitement. Nous devons donc les ignorer. Seulement, ces pôles prennent la place de pôles complexes, et le classement par partie réelle décroissante s'en retrouve affecté.
- L'ordre. Nous classons les pôles pour les regrouper par argument. Cela marche dans la mesure où chaque intervalle  $[\theta_1, \theta_2]$  ne contient des pôles que d'un seul ordre. Or, c'est rarement le cas. De plus, les racines réelles que nous avons ignorées ont souvent une partie imaginaire

<sup>4</sup>De partie imaginaire quasi nulle

faible, mais non nulle. Elles s'intercalent donc au milieu des racines complexes et décalent toutes les valeurs.

Nous avons développé un algorithme qui permet de résoudre ces trois difficultés. Celui-ci

- Calcule et classe les pôles par partie réelle, en excluant ceux dont le module n'est pas proche de 1
- Calcule les barycentres et des écarts types pour chaque groupe ainsi représenté (voir figure 4.10)
- Considère comme faisant partie du groupe de pôles  $i$  les pôles se trouvant à une distance proportionnelle à l'écart-type du barycentre des pôles  $i$ .

Malgré cela, les résultats sont demeurés médiocres. Les barycentres trouvés n'étaient pas assez proches pour une même voyelle, et la reconnaissance devenait aléatoire.

De plus, le manque d'importance donné par cette méthode à l'atténuation relative des différentes fréquences nous a poussés à continuer dans une autre voie.

## Chapitre 5

# Applications : le mot de passe vocal et la calculatrice vocale

### 5.1 Mot de passe vocal

Le but de ces algorithmes était au départ de vérifier l'identité d'un locuteur et le mot qu'il prononce. Voici alors comment on agence les différents programmes.

1. Chaque locuteur enregistre les voyelles principales ("a", "e", "i", "o", "u"), et les voyelles annexes ("on", "ou", "an", "un", "é", "è"). Ces voyelles vont constituer le dictionnaire.
2. On sauvegarde cette base.
3. Ensuite, avec le son à tester, on sépare les syllabes arbitrairement suivant les proportions observées lors de l'enregistrement du mot de passe.
4. Pour chaque syllabe, on calcule la réponse fréquentielle du filtre LPC associé grâce à `traitementLPC`.
5. Puis, on calcule la distance de cette réponse. On stocke la lettre la plus probable pour chaque personne, ainsi que l'assurance (si la personne 1 avait prononcé ce mot, ce serait un ... ; si la personne 2 avait prononcé ce mot, ce serait un ... ; *etc*).
6. A la fin de chacune des reconnaissances, on renvoie la personne dont la somme des assurances est maximum, ainsi que les lettres qui lui sont attribuées.

On constate qu'en fait de reconnaissance de mot, il s'agit plus ici de reconnaissance des voyelles balisant le mot. Cela est d'autant plus vrai que :

- Les consonnes mettant en place un régime stationnaire, elles diminuent l'efficacité de l'algorithme
- La reconnaissance ne reconnaît pas à proprement parler une voyelle, mais la voyelle la plus proche
- Les consonnes pourraient être utilisées afin de séparer les voyelles, mais nous n'avons pas eu le temps d'implémenter cette fonctionnalité. La séparation des syllabes se fait donc de façon manuelle grâce à une référence (voir paragraphe 4.2.3)

### 5.2 La calculatrice vocale

#### 5.2.1 Le principe

Lors des dernières semaines du ModEx, nous avons essayé d'utiliser nos algorithmes pour un autre programme : la calculatrice. Dans le principe, la calculatrice vocale est assez peu différente du mot de passe vocale. Il y a cependant deux différences majeures :

- La reconnaissance du locuteur est inutile
- La structure du dictionnaire est assez différente, celui-ci pouvant contenir des mots de plus d'une syllabe ou à syllabes diphtonguées par exemple.

La calculatrice vocale est basée sur la reconnaissance de lexèmes<sup>1</sup>. Le dictionnaire doit donc contenir tous les lexèmes afin de reconnaître les syllabes prononcées. Une fois les lexèmes reconnus,

---

<sup>1</sup> un lexème est une entité grammaticale d'un programme. Ici, il s'agit d'un nombre (par exemple "1" ou "2") ou d'un opérateur (par exemple "+" ou "-")

il faut passer à l'analyse syntaxique, qui doit interpréter cette suite de lexèmes. Pour des raisons de temps, et parce que cela avait peu de rapport avec notre objectif de départ du ModEx, cela a été fait directement par la commande `eval` de MatLab.

Cependant, après coup, nous pouvons dire que l'analyse syntaxique aurait pu nous aider à améliorer nos résultats. En effet, celle ci aurait

- Réduit le nombre de mot possible à certains endroits de la phrase (par exemple, une formule mathématique ne commence pas par un opérateur)
- Géré les éventuelles erreurs et essayé de les rattraper.

Nous avons donc essayé de faire une micro-analyse syntaxique lors du dernier cours, afin d'obliger le programme à reconnaître alternativement un nombre et un opérateur.

### 5.2.2 L'implémentation et les résultats

Le programme est le même que pour la reconnaissance de mots. Là encore, pour des raisons pratiques, la séparation des mots et des silences s'est faite par la méthode la plus simple possible. Le dictionnaire contenait 8 mots :

Les chiffres	Les opérateurs
1	+
2	-
3	*
6	
7	

Cependant, les résultats ont été moins bons que ceux de la reconnaissance de mots pure. Certaines formules mathématiques ont tout de même été reconnues totalement.

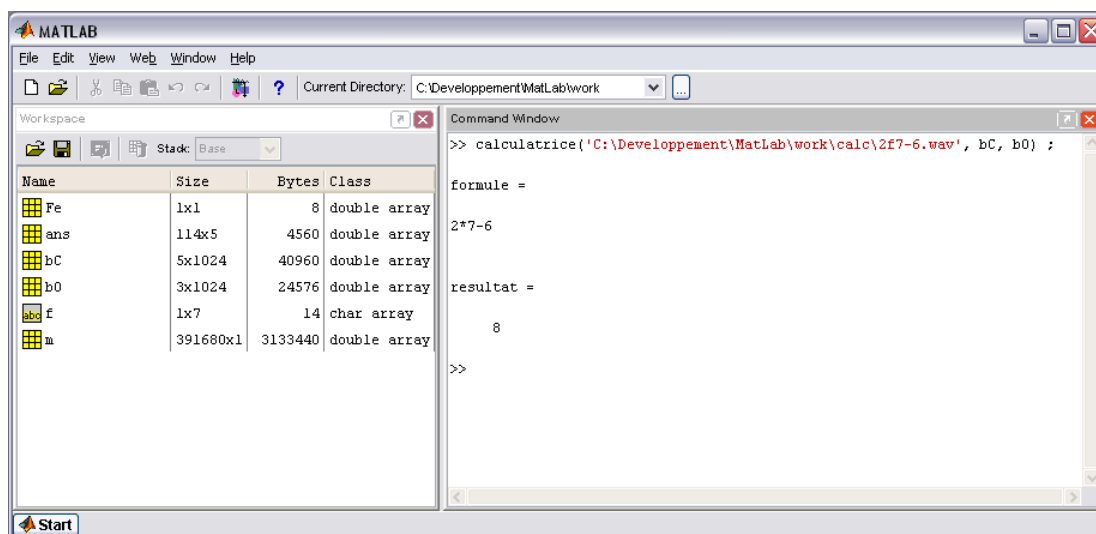


FIG. 5.1 – Un exemple d'utilisation de la calculatrice

### 5.2.3 les problèmes spécifiques à la calculatrice

Le principal problème de la calculatrice est son dictionnaire. En effet, celui-ci présente deux caractéristiques qui le rendent difficile à utiliser

- Les mots de ce dictionnaire ont des voyelles très souvent diphtonguées (par exemples tro-is, mo-ins)
- Les mots du dictionnaire sont souvent phonétiquement très proches : un et moins, trois et quatre, six et dix. C'est pour cette raison que nous n'avons un dictionnaire que de cinq chiffres.

Afin de contourner ces problèmes, nous avons eu recours à deux solutions en parallèle :

- La séparation des dictionnaires : avec deux dictionnaires, un pour les opérateurs et un pour les chiffres, la reconnaissance est plus facile et plus sûre.

– Une nouvelle distance

Cette distance est spécifique au cas de la calculatrice, et s’est avérée assez efficace.

1. Le dictionnaire n’est pas composé des mots, mais de la voyelle caractéristique du mot :
  - e pour 2
  - a pour 3 et \*
  - ...
2. La formule vocale est d’abord séparée en mots
3. On divise chacun des mots en fenêtres de  $23ms$
4. Sur chaque fenêtre, on effectue la reconnaissance en conservant la distance  $d$  au plus proche élément du dictionnaire.
5. Pour les cinq fenêtres les plus proches, on ajoute au score de l’entrée du dictionnaire reconnue  $\frac{1}{d}$
6. Le plus grand score donne le mot

Cette distance a permis de passer de 2 chiffres et 1 opérateur à 5 chiffres et 3 opérateurs. Il est toutefois intéressant de remarquer que dans le cas du mot de passe vocal, et dans l’esprit où nous avons traité celui-ci, elle serait totalement inefficace. En effet, dans le cas du mot test “cacao”, et comme la séparation des syllabes est imparfaite et arbitraire, il serait fréquent que la découpe du mot mêle du “a” au “o” de la dernière syllabe. Si la reconnaissance se fait mieux pour le “a” que pour le “o” sur trois fenêtres, la syllabe sera reconnue comme un “o”.

Le principe de cette distance pourrait tout de même servir car il permet de ne pas tenir compte des instabilités des consonnes qui mettent souvent le son en régime instable et transitoire.

# Chapitre 6

## Conclusions et remarques

### 6.1 Ce qu'on aurait pu faire de plus

Le ModEx se fait sur une période très courte, et malgré le temps investi en classe mais aussi chez soi (peut-être 50% du travail), il n'y a pas le temps d'explorer toutes les pistes à envisager. Si nous continuions notre projet<sup>1</sup>, nous irions dans ces directions :

- Utiliser la séparation voix/non-voix pour la reconnaissance des mots. Cela demanderait la programmation d'un algorithme destiné à recoller les syllabes, ce qui n'est pas évident mais a priori surmontable.
- Amélioration de la calculatrice : ajout de nombres, d'opérateurs, et surtout analyse syntaxique qui ouvrirait la porte aux nombres à plusieurs chiffres
- Nouvel essai de la méthode des pôles et des centroïdes, abandonnée faute de temps, et par nécessité de résultats. Cette méthode et les défis informatiques qu'ils représentent nous on paru intéressant<sup>2</sup>
- On aurait pu vérifier la qualité du dictionnaire : par exemple, on aurait pu demander à la personne qui a enregistré ces échantillons de répéter ces voyelles et vérifier que les résultats sont corrects. Si tel n'est pas le cas, on demande à la personne de réenregistrer les voyelles mal "perçues". (En effet, on a pu vérifier tout au long de nos tests que soit un échantillon de base est correct, soit il ne fonctionne presque jamais). Une autre méthode est de demander à la personne d'enregistrer quatre fois les mêmes voyelles et en les comparant entre elles, on détermine quelle est la base qui donne le meilleur résultat.
- La séparation des syllabes aurait pu tirer profit de l'état chaotique entraîné par les consonnes. Une détection d'un tel état permet de séparer plus facilement les syllabes, tout en laissant le problème du passage en fondu (le "ao" de "cacao").
- Enfin, le principal défaut de ce programme est qu'il n'identifie pas une voyelle, mais une probabilité d'être une voyelle. Il aurait fallu affecter une distance maximum au delà de laquelle la syllabe n'est plus reconnue. Car on arrive à des situations dénuées de sens : essayer à tout prix de reconnaître un "o" sur un dictionnaire ne contenant que des "a" et des "e" entraîne une baisse d'efficacité du programme.

### 6.2 Le ModEx

Nous ne voulons juste, par ce paragraphe, donner un avis d'élève qui permettra peut-être de faire évoluer ce module souvent controversé.

Le ModEx nous a paru une bonne façon de nous attaquer à un projet, aussi modeste soit-il. Il nous a en outre apporté un complément par rapport au cours d'informatique fondamentale qui peut sembler parfois trop théorique.

Étant tous les deux attirés par des études (quatrième année) dans le domaine de l'informatique et des technologies de l'information, il nous a semblé utile et profitable de voir concrètement de quoi il s'agissait.

---

<sup>1</sup> Et nous le continuerons peut-être !!

<sup>2</sup> et rester sur un échec n'est pas satisfaisant !

## 6.3 Conclusion

Ce à quoi nous sommes arrivés dans ce projet n'est qu'un début. L'utilisation de méthodes éprouvées nous a donné des résultats honorables sans être parfaits.

Plus qu'à apporter à la technologie par ce travail, nous avons cherché à en retirer le maximum du point de vue pratique. Sur ce point, les résultats sont très positifs. Il nous reste tout de même en plus la satisfaction de la paternité d'un programme qui reste tout de même efficace.