***ISEN-Toulon Méditerranée***  ***Semestre 6 - 2024***

Une image contenant Police, Graphique, texte, graphisme

Description générée automatiquement

Compte rendu du  
projet d’IA / THS  
du S6

*Projet réalisé par CRAPE Amaury et BARBAUD Malo*

Sommaire :

[1 - Introduction 2](#_Toc168838729)

[2 - Algorithme de Cooley-Tuckey (FFT) 2](#_Toc168838730)

[3 – Neurones 6](#_Toc168838731)

[a) Habilitation et test de neurones. 6](#_Toc168838732)

[b) Travail sur la résilience des neurones 8](#_Toc168838733)

[4 – Assemblage Neurones-FFT 8](#_Toc168838734)

[a) Traduction Signal vers 8](#_Toc168838735)

# 1 - Introduction

Le projet d’IA de ce semestre consiste en la programmation de neurones capables de reconnaître des sons qu’on leur présente. En vue de réaliser cet objectif, nombres de documents et segments de codes nous sont fournis. Parmi eux, une implémentation de la Transformée de Fourier (qui nous permettra de décomposer les sons en fréquences), une interface INeurone et ses implémentations qu’il faudra compléter pour faire fonctionner le réseau ainsi qu’une banque de sons et un algorithme d’échantillonnage.

Ce rapport a pour but de présenter le déroulement du projet : nos observations, nos raisonnements, nos expériences et nos résultats.

# 2 - Algorithme de Cooley-Tuckey (FFT)

Le son est une onde : il correspond à une vibration dans l’espace en fonction du temps. Pourtant, quand il parvient à nos oreilles, nous entendons bien une note, c’est à dire une fréquence, et non une vibration. L’oreille mesure donc le poids relatif de chaque fréquence dans le signal temporel : c’est ce qu’on appelle une transformée de Fourier.

Seulement, l’oreille opère à partir d’un signal analogique, un ordinateur n’a pas ce luxe : Pour pouvoir traiter une information celle-ci doit-être discrétisée, c’est à dire échantillonnée.

Nous travaillerons donc avec une transformée de Fourier discrète (TFD), l’équivalent numérique de la transformée de Fourier vue en cours.

Dans le code d’exemple de la TFD fourni, nous démarrons par appliquer une fonction transformée de Fourier sur un signal réel défini par :

Avec P = 1 et = 16.

Une image contenant texte, capture d’écran, Police, typographie

Description générée automatiquement

Bien que P et soient modifiables, nous sommes restés dans un premier temps avec les valeurs initialisées par défaut pour commencer à analyser son comportement.

On remarque que la fonction FFT renvoie une série de points complexes, décrits en coordonnées cartésiennes et polaires, censés représenter les pics de fréquence perçus dans le signal traité.

Tracer ces points sur un logiciel comme *Geogebra* ne laisse pas apparaître d’informations particulières : mêmes si une certaine pente semble se dessiner si on multiplie la plupart des points par 10^16, il n’y pas de quoi en tirer des conclusions.

Une image contenant texte, capture d’écran, diagramme, nombre

Description générée automatiquement

Le signal décrit n’étant composé que d’un simple cosinus, on devrait idéalement apercevoir des pics de fréquence autour de et , c’est-à-dire l’unique fréquence qui compose le cosinus. Si on se ramène au modèle du signal vu dans le cours :, alors le du signal traité vaudrait , soit 1/16, pourtant aucune des valeurs acquises n’approchent cet ordre de grandeur. Cela est probablement dû à un phénomène de repliement spectral : l’implémentation ici-faite de l’algorithme ne permet pas de réel contrôle sur la fréquence d’échantillonnage de notre signal, ainsi la préservation de l’information n’est pas garantie par le théorème de Shannon.

En réhaussant la fréquence (jusqu’à P = 3 par exemple), on peut cependant observer que les valeurs se resserrent et gagnent en amplitude

Une image contenant texte, capture d’écran, Police, menu

Description générée automatiquement

Etant donné que l’on prend toujours le même nombre de points, ce resserrement s’explique assez naturellement par l’augmentation de la fréquence du signal. L’origine de cette croissance perçue des valeurs est moins certaine : elle peut être due au repliement spectral évidemment, mais on peut également conjecturer qu’avec la croissance de la fréquence, les 16 points pris par l’échantillon sont « à un autre endroit » sur la courbe, et effectivement, si on en croit les coordonnées cartésiennes des points, les abscisses réelles ne sont pas les mêmes que celles précédemment acquises.

Si à la place du cosinus, nous appliquons un sinus, on remarque que certaines données semblent s’inverser : on retrouve des points d’ordonnée -8, 0, ou 8, qui auparavant étaient en abscisse.

Exemple ici avec P = 1 et = 16

Une image contenant texte, capture d’écran, Police, typographie

Description générée automatiquement

(Lien trigonométrique avec le déphasage de pi/2 ?)

# 3 – Neurones

1. Habilitation et test de neurones.

Une fois la fonction *apprentissage* complétée nous testâmes son efficacité sur les fonctions *ET* puis *OU* grâce à la class *NeuroneHeaviside* fournie.

Donnés récupérées pour la fonction *OU*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Moyenne Poids 1 OU** | **Poids 1** | **Poids 2** | **Biais** | **Nb Tours** |
| 0,398310533 | 0,21399641 | 0,7053076 | -0,07862749 | 3 |
| **Moyenne Poids 2 OU** | 0,2008726 | 0,12676454 | -0,11110028 | 22 |
| 0,284518954 | 0,13093108 | 0,07312061 | -0,045089677 | 25 |
| **Moyenne Biais OU** | 0,04311706 | 0,41229948 | -0,033208832 | 18 |
| -0,121773466 | 0,82820696 | 0,13800931 | -0,09897482 | 14 |
| **Moyenne nb Tours OU** | 0,24553373 | 0,22162768 | -0,1810162 | 26 |
| 14,77777778 | 0,9341661 | 0,65893525 | -0,42317083 | 1 |
|  | 0,21535626 | 0,13719103 | -0,040588275 | 9 |
|  | 0,7726146 | 0,08741509 | -0,08418479 | 15 |

Donnés récupérées pour la fonction *ET*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Moyenne Poids 1 ET** | **Poids 1** | **Poids 2** | **Biais** | **Nb Tours** |
| 0,328841146 | 0,30550325 | 0,13597363 | -0,38373792 | 23 |
| **Moyenne Poids 2 ET** | 0,33438855 | 0,2022686 | -0,4677974 | 26 |
| 0,24362746 | 0,18793559 | 0,6677758 | -0,8114728 | 12 |
| **Moyenne Biais ET** | 0,5196526 | 0,1873945 | -0,6775151 | 15 |
| -0,499510358 | 0,38299108 | 0,1338371 | -0,42096865 | 18 |
| **Moyenne Nb Tours ET** | 0,3159633 | 0,07027962 | -0,35877842 | 26 |
| 20 | 0,32325828 | 0,15006164 | -0,42198026 | 31 |
|  | 0,27519792 | 0,38855198 | -0,48095652 | 1 |
|  | 0,31467974 | 0,25650427 | -0,47238615 | 28 |

On observe que les poids finaux de chaque test sont drastiquement différents, cela peut être expliqué par la façon d’initialiser les poids. En effet, à l’initialisation, les poids (tout comme le biais) ont un poids aléatoire variant de -1 à 1.

Ainsi, la fonction *apprentissage* ayant tendance à faire varier les poids et biais et s’arrêter dès que les conditions sont remplies, elle ne procède à aucune harmonisation des poids comme un humain serait susceptible de le faire.

1. Travail sur la résilience des neurones

Pour tester les réactions face au bruitage, nous avons codé une fonction permettant de bruiter indépendamment les entrées de .

Pour ce qui est des fonctions *ET* et *OU*, les neurones de type Heaviside ne sont pas incapacités même avec un bruitage de 0,5.

# 4 – Assemblage Neurones-FFT

1. Traduction Signal vers

Nous avions essayé de traiter les informations de la FFT via l’utilisation du module. Cependant, bien que les informations étaient suffisantes, nous avons remarqué une perte d’informations lié à cette méthode.

Entrées bruitées ; traiter les infos (FFT) faire par le module \*\* essayé par module mais perte d’info (suffisant mais pas total) ; échantillon de 512 points ; Le neurone en ET résiste très bien au bruit.