

|  |
| --- |
|  |
| Compte rendu –  Projet L3  Outils maths |
| Transformée de Fourier |

|  |
| --- |
| Tom BARBIER – Brandon PERE  Année 2022 - 2023 |

Table des matières

[I. Introduction 3](#_Toc118820687)

[II. Modélisation de la tortue **Erreur ! Signet non défini.**](#_Toc118820688)

[1. La carapace **Erreur ! Signet non défini.**](#_Toc118820689)

[2. Les nageoires **Erreur ! Signet non défini.**](#_Toc118820690)

[3. La tête **Erreur ! Signet non défini.**](#_Toc118820691)

[III. Gestion des textures **Erreur ! Signet non défini.**](#_Toc118820692)

[IV. Gestion des animations **Erreur ! Signet non défini.**](#_Toc118820693)

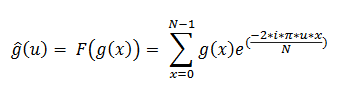
[V. Gestion des lumières **Erreur ! Signet non défini.**](#_Toc118820694)

[VI. Conclusion **Erreur ! Signet non défini.**](#_Toc118820695)

# Introduction

Nous allons voir comment et ce à quoi sert la transformé de Fourier, ainsi que la  
transformé de Fourier rapide. Nous allons expliquer pourquoi nous évoquons les deux. Il faut savoir que la transformé de Fourier est très utile, cependant, elle est difficilement utilisable lorsque nous l’implémentons, pourquoi ? En raison de sa complexité. En effet, la complexité de la transformé de Fourier est de 𝑂(𝑁²), c’est pourquoi il existe, la transformé de Fourier rapide, qui a une complexité simplifiée. En effet, sa complexité est de 𝑂(𝑁 ∗) pour des tableaux de tailles de taille . Dans un premier temps nous construirons la fonction pour des tableaux à 1 dimension. Puis nous généraliserons à deux dimensions en ce reposant sur la 1D.

# Transformée de Fourier Direct



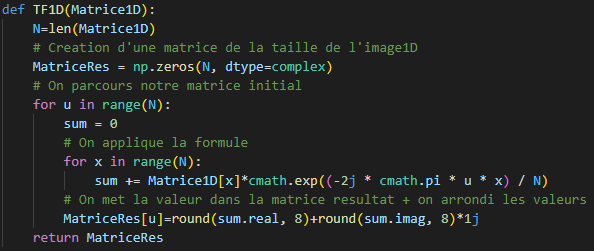
u index du tableau

N taille du tableau

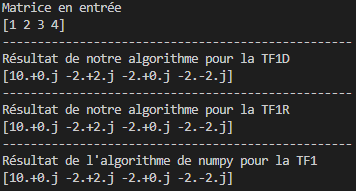
g tableau source

ĝ tableau de destination

On remarque donc que pour chaque élément nous devons parcourir l’entièreté du tableau source ce qui explique sa complexité de O(N²). En l’implémentant en python cela nous donne :

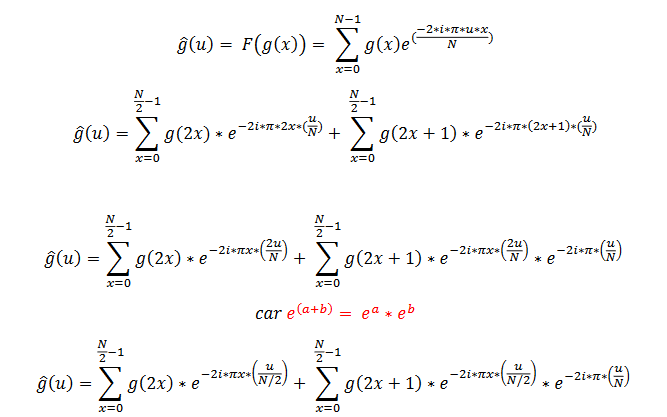


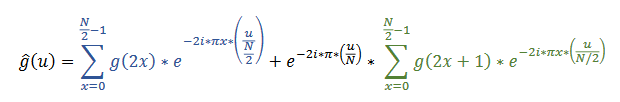
On peut comparer les résultats avec la fft de numpy. (bibliothèque que nous utilisons pour simplifier le code)



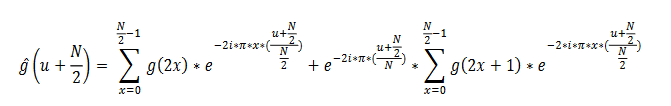
# Optimisation

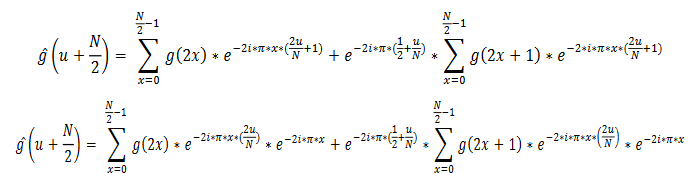
Nous pouvons séparer la somme en deux sommes afin d’avoir les éléments pairs d’un coté et impairs de l’autre côté ce qui donne :

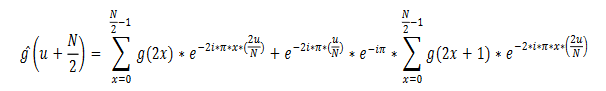
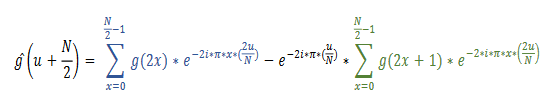


Le terme ne dépend pas de la variable de la somme il est donc possible de la sortir de la somme :

Nous avons donc en bleu à gauche les éléments pairs et en vert à droite les éléments impairs. On a donc notre tableau, mais celui-ci n’est pas complet car le tableau est de taille N/2.

Nous devons donc calculer la seconde moitié soit :

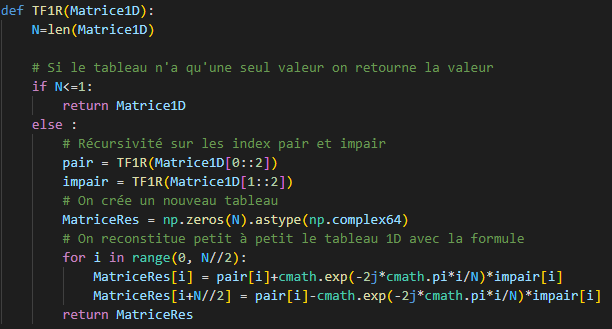
Simplifions cette expression :



= = 1 avec

=

Maintenant que nous avons simplifier le problème nous pouvons passer à l’implémentation. L’avantage de la représentation mathématique que nous avons est qu’elle laisse bien voir un algorithme récursif.



On peut comparer les résultats avec la fft de numpy.

