

Réalisé par:

TP 2 : AC

TAPE-WALI Z. B. Junior A.

TRAORE Zié Amara

Année académique : 2022 - 2023

*ESIR 2 SI*

Compte rendu du TP d’Algorithme et Complexité

**SOMMAIRE**

[SOMMAIRE 1](#_Toc325698508)

[I. Première partie 2](#_Toc201439511)

[a) Exercice 1 2](#_Toc717934757)

[b) Exercice 2 2](#_Toc1620640438)

[c) Exercice 3 3](#_Toc843083870)

[d) Exercice 4 3](#_Toc1114675390)

[e) Exercice 5 4](#_Toc684934613)

[II. Deuxième partie 5](#_Toc1455413921)

[a) Énoncé 5](#_Toc426628915)

[1. Exercice 1 5](#_Toc1669052342)

[2. Exercice 2 7](#_Toc2119823881)

[3. Exercice 3 7](#_Toc1981050227)

[4. Exercice 4 9](#_Toc24992490)

[b) Demo Matlab 9](#_Toc1922929339)

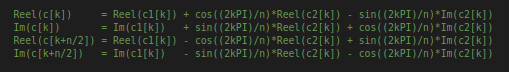
.

1. **Première partie**
2. **Exercice 1**

Pour l’implémentation de la fonction *CpxTab combine(CpxTab c1, CpxTab c2), e*n utilisant les instructions données dans l’énoncé on pose c la combinaison de c1 et c2 selon la formule vue en TD

après calcul on obtient :

Vous trouverez l’implémentation de cette fonction dans le fichier « FFT\_1D.java » .



1. **Exercice 2**

En utilisant la fonction *CpxTab combine(CpxTab c1, CpxTab c2),* nous avonsécrire une version récursive (***CpxTab FFT(CpxTab x)***) du calcul des coefficients de Fourier.

Nous avons par la suite tester notre proposition de code sur l’exemple de l’exercice 5 du TD3. C’est à dire le tableau de double t = {1, 2, 3, 4}.

L’image suivante vous montre le résultat de ce test que nous avons afficher grâce à la méthode *toString()* de la classe CpxTab().



1. **Exercice 3**

Nous avons proposer dans le fichier « FFT\_1D.java » une implémentations de la fonction ***CpxTab FFT\_inverse(CpxTab y)***, qui retrouve le signal d’entrée à partir de ses coefficients de Fourier.

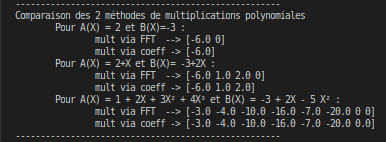
L’image suivante est le résultat du test de notre implémentations sur la TFD qu’on a calculer à l’exercice 1 :

1. **Exercice 4**

Nous avons compléter la fonction *CpxTab multiplication\_polynome\_viaFFT* qui réalise la multiplication de polynômes en utilisant la transformée de Fourier, selon la méthode vue en en TD.

Vous trouverez l’implémentation dans le fichier « FFT\_1D.java » .

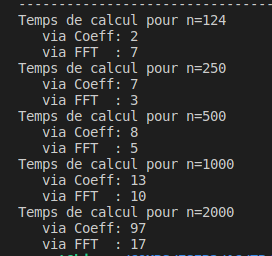
L’image suivante est le résultat des tests faits avec code qu’on a proposé.



1. **Exercice 5**

En utilisant le code présent dans le main, nous avons calculé pour différentes tailles de signal d’entré, le temps d’exécution de ces deux méthodes de multiplication de polynôme.

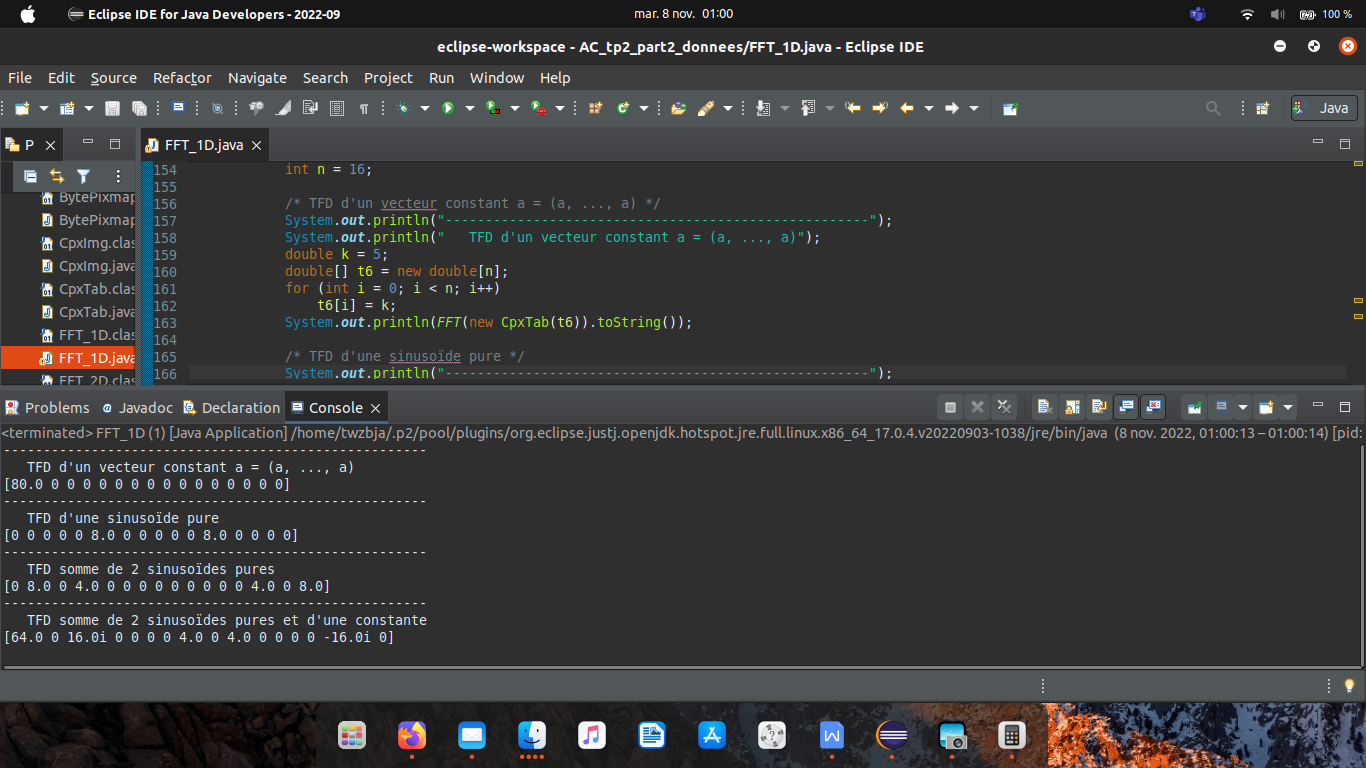
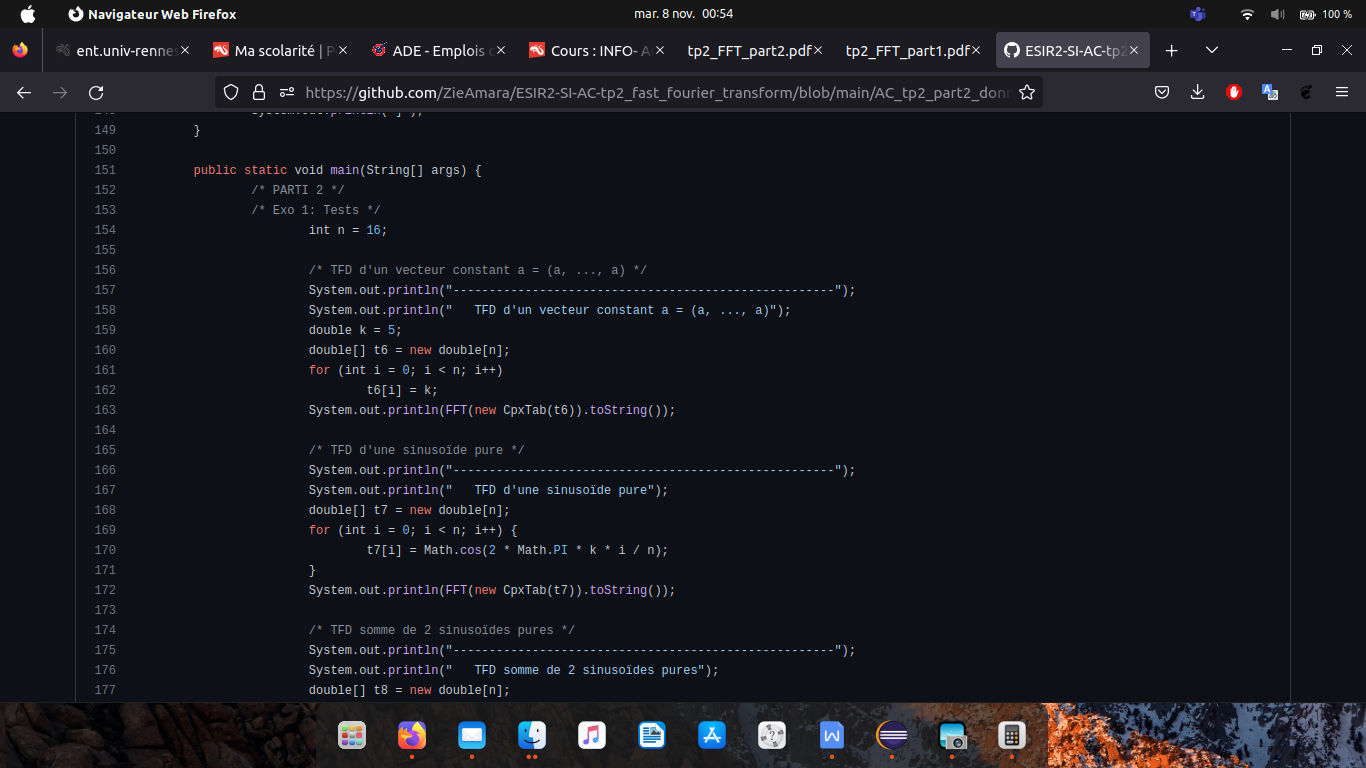
L’image suivante est le résultat de test du code :



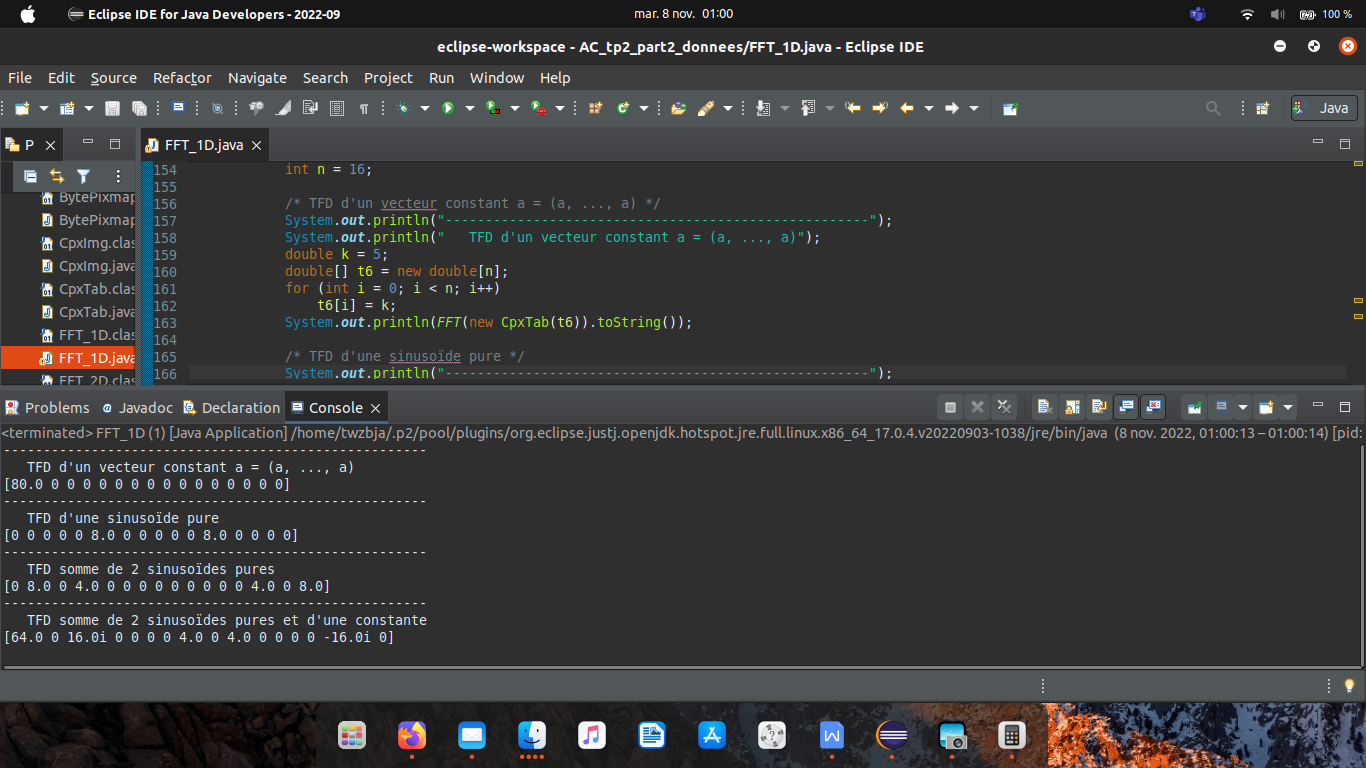
Nous constatons en faisant croître la valeur que le temps de calculs par la FFT est meilleur que le temps de calculs par les coefficients dès lors que la taille du signal devient importante.

1. **Deuxième partie**
2. **Énoncé**
3. **Exercice 1**

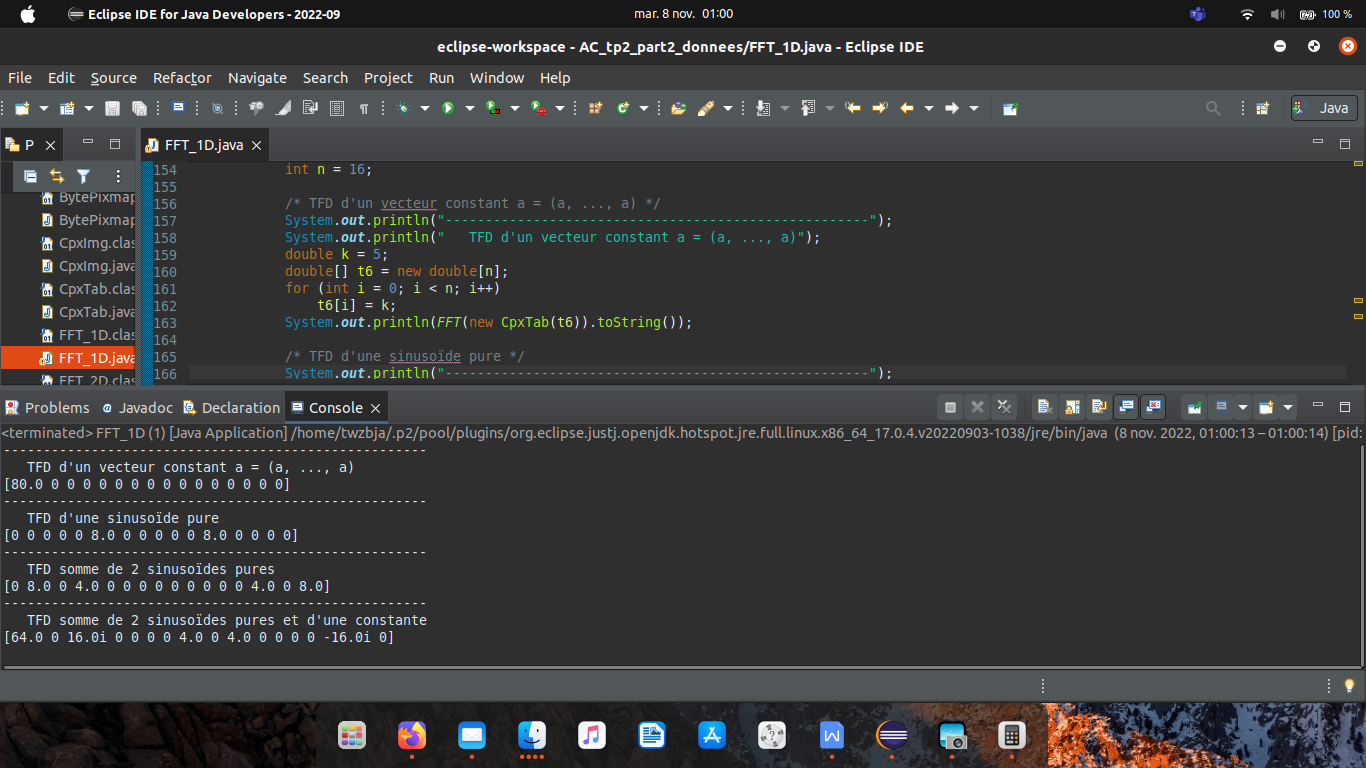
* Présentation du code et du résultat obtenu pour la TFD d'un vecteur constant a = (a, ..., a)

****

* Présentation du code et du résultat obtenu pour la TFD d'une sinusoïde pure : **a =**

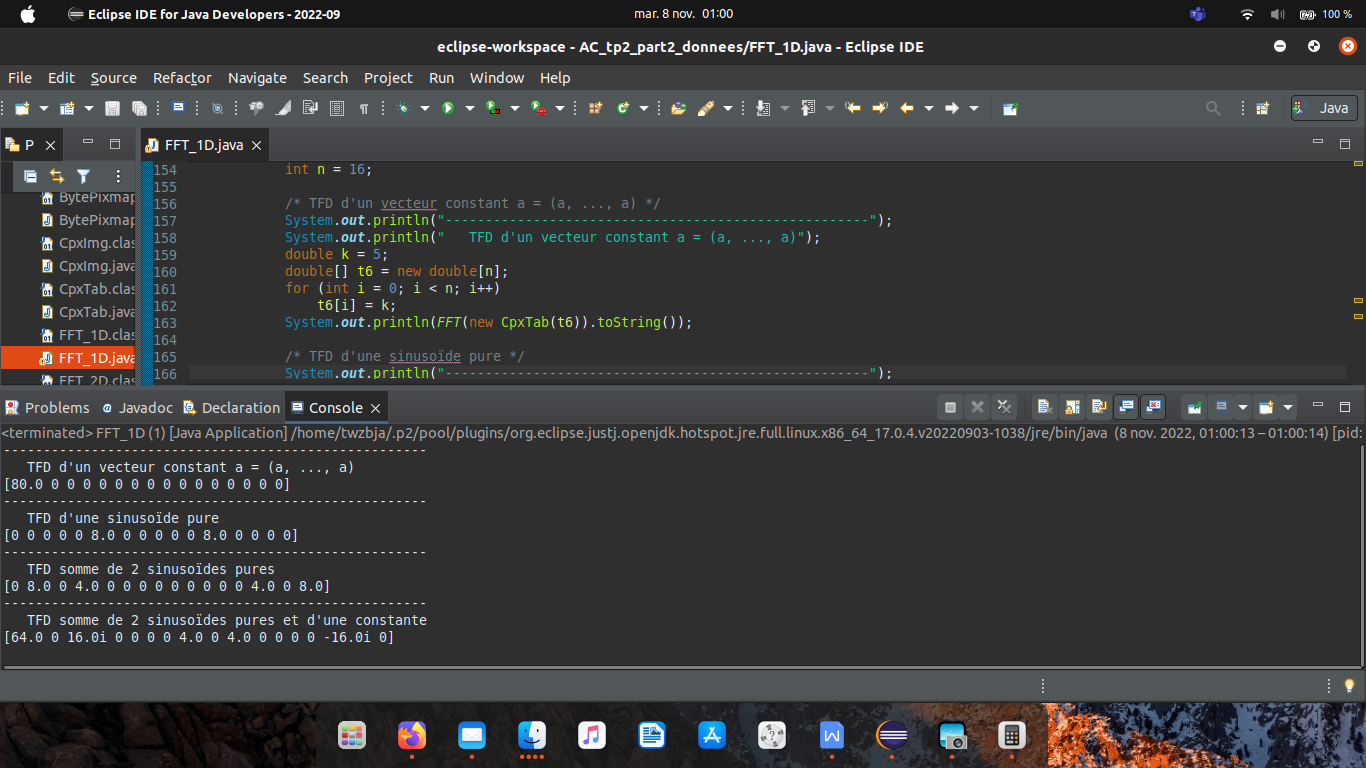
****

* Présentation du code et du résultat obtenu pour la TFD de la somme de deux sinusoïdes pures : **a = (a0, . . . , an−1)**

****

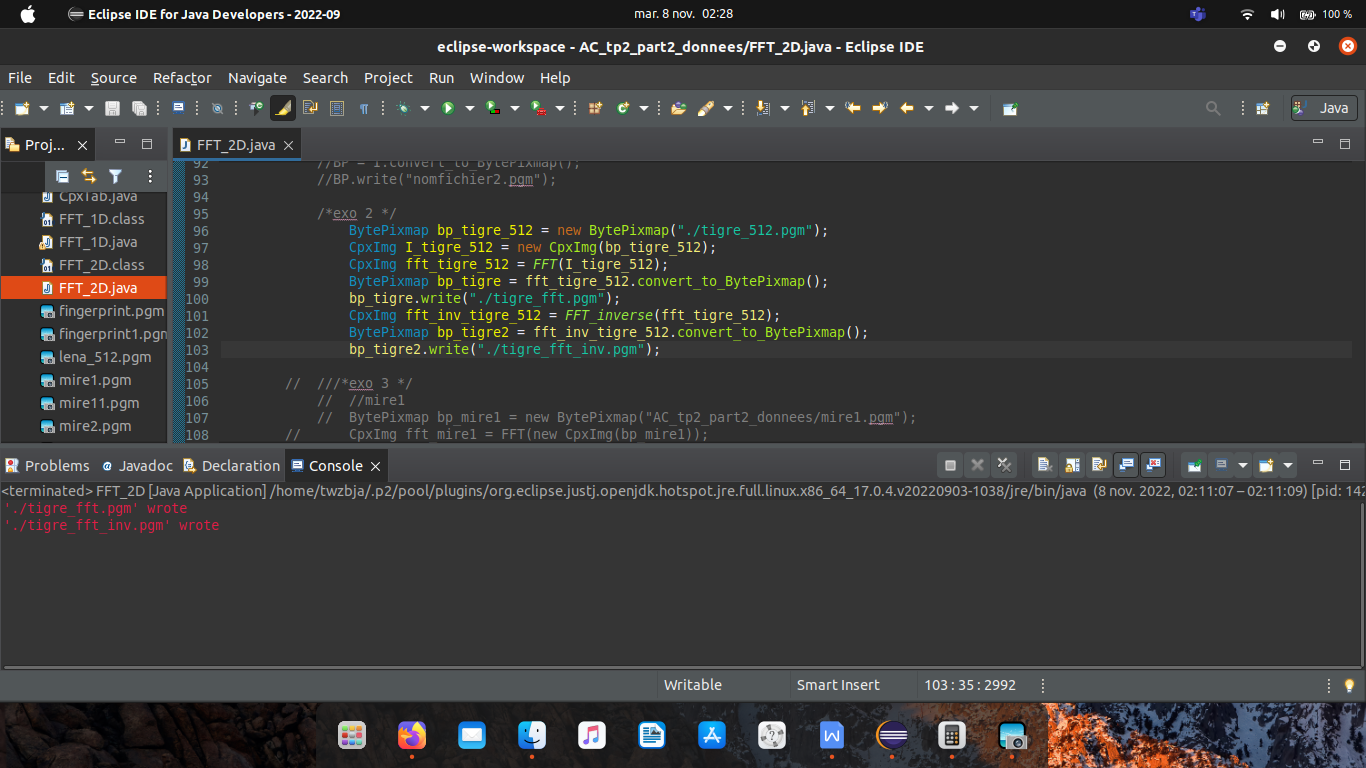
* Présentation du code et du résultat obtenu pour la TFD de la fonction **a = (a0, . . . , an−1)**

****

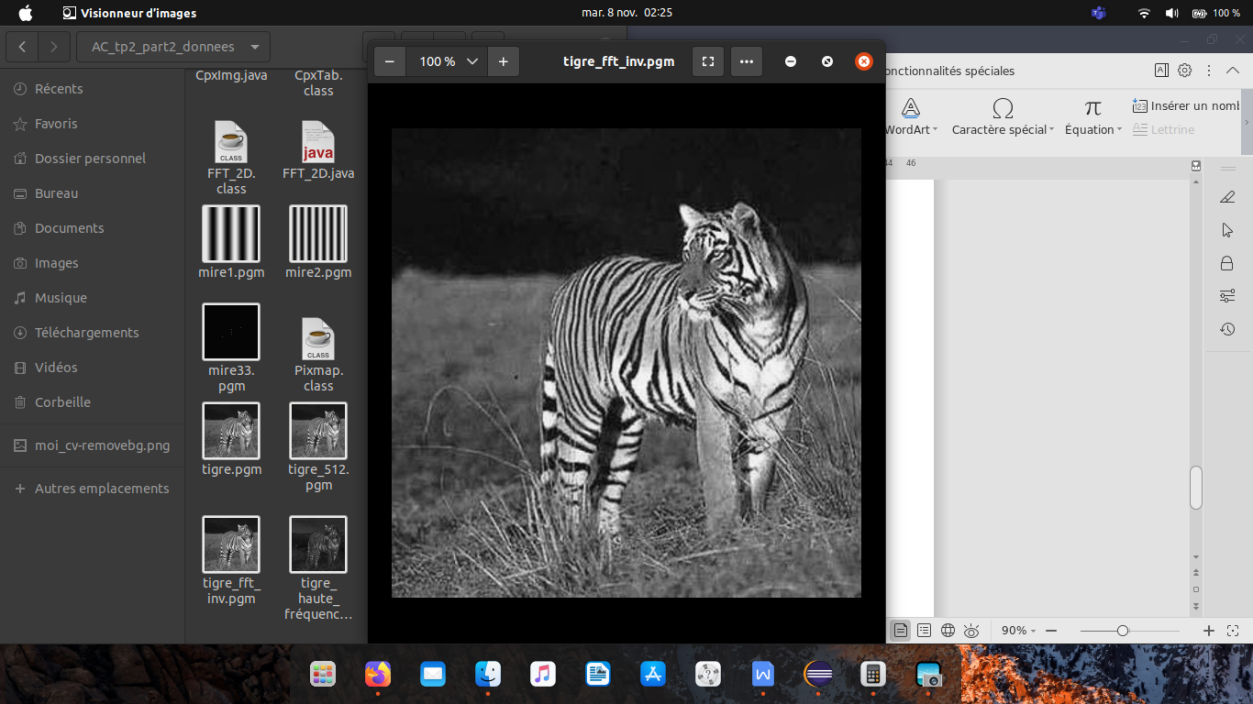
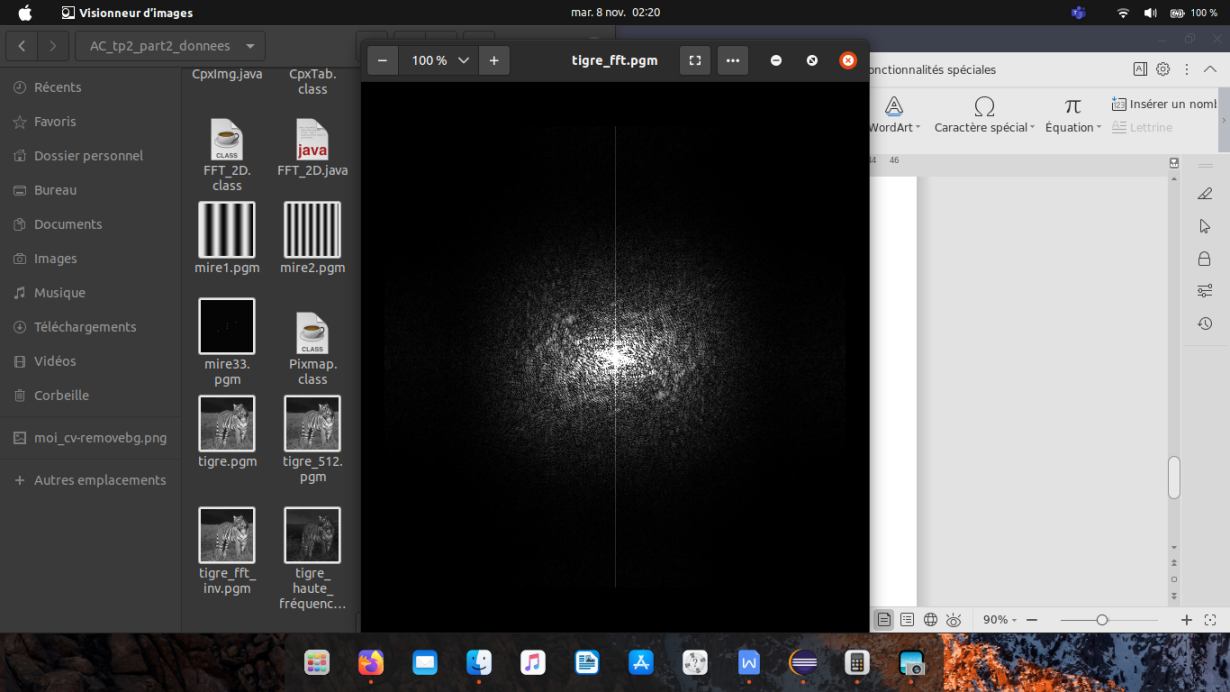
****

1. **Exercice 2**

La figure suivante présente les lignes de codes pour la vérification du bon fonctionnement des fonctions des implémentations 1D, par le calcul du TFD de l’image tigre\_512.pgm :



Lorsque nous appliquons la FFT sur l’image tigre\_512.pgm puis la FFT inverse nous obtenons la même image que celle de tigre\_512.pgm .

****

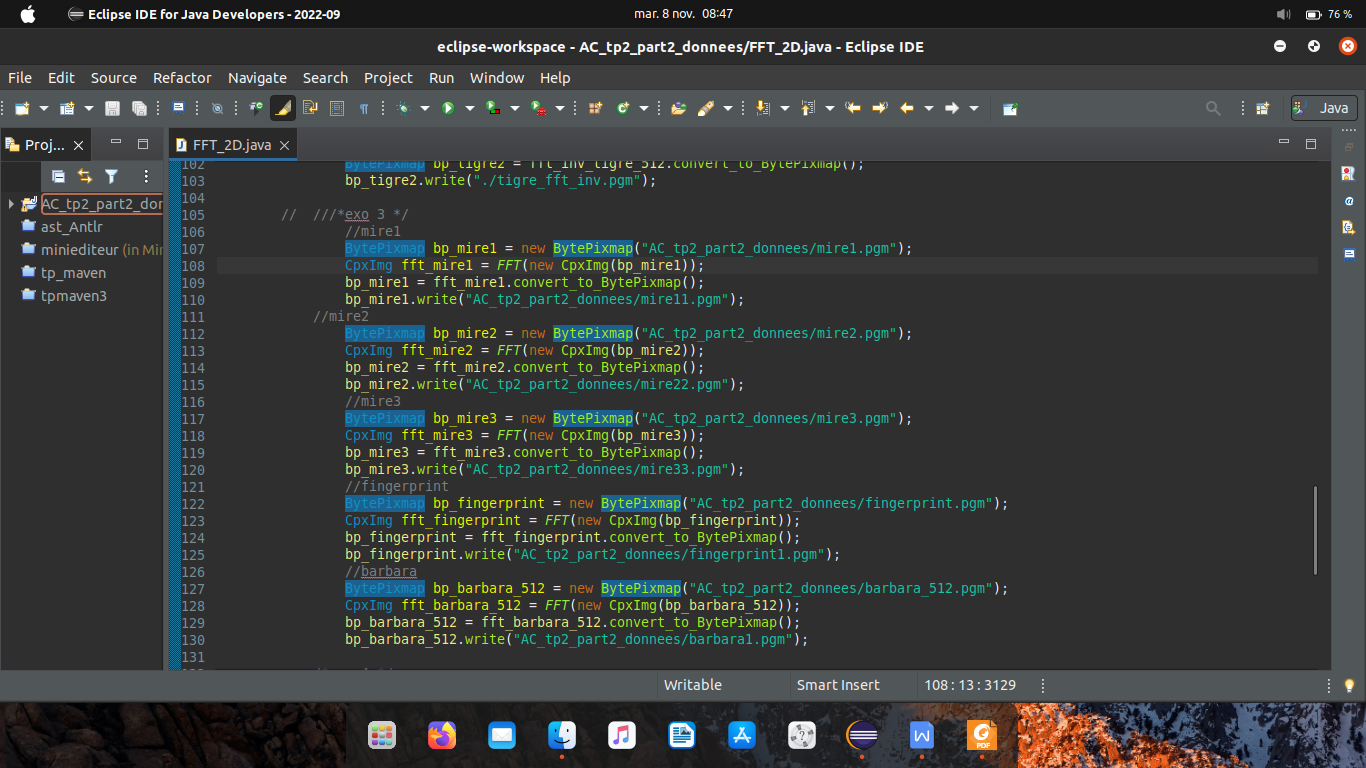
**Image :** TFD sur tigre\_512.pgm

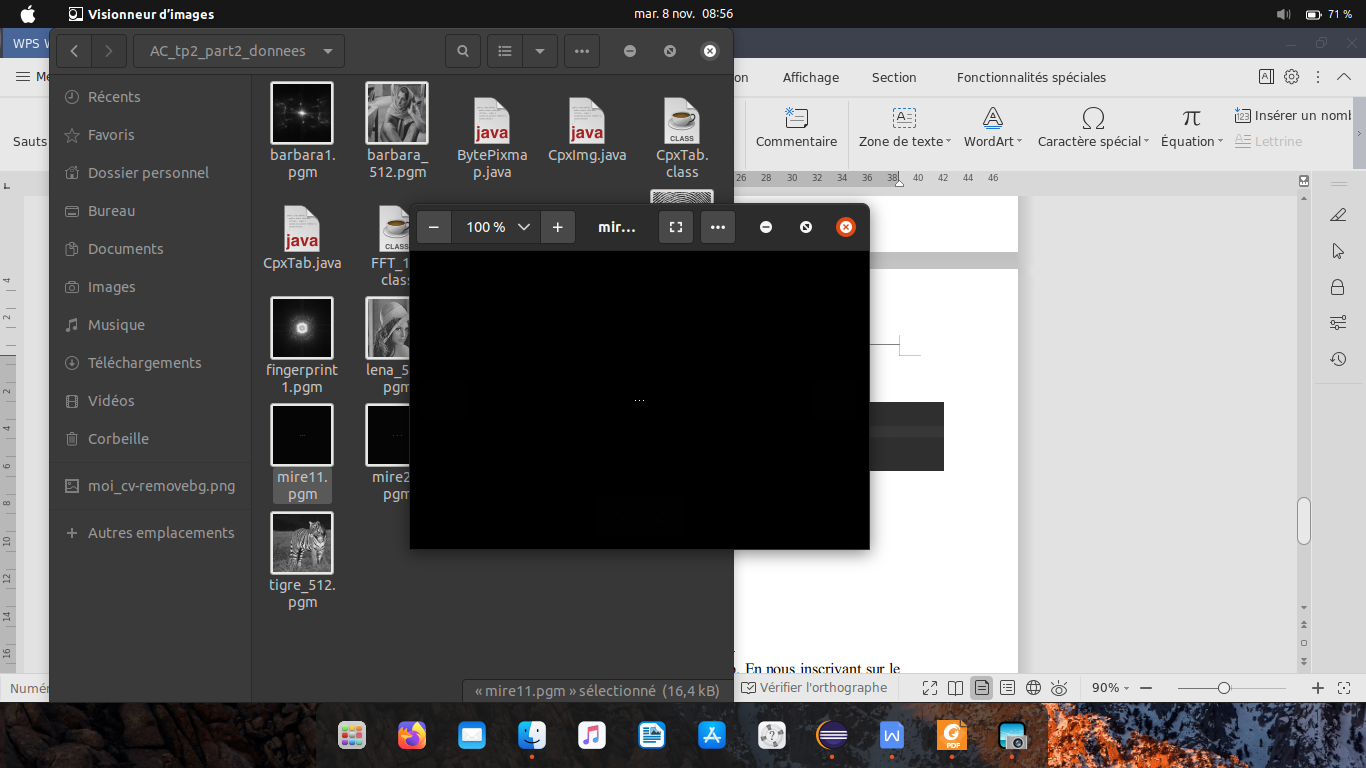
**Image :** TFD et TFD inverse sur tigre\_512.pgm

1. **Exercice 3**

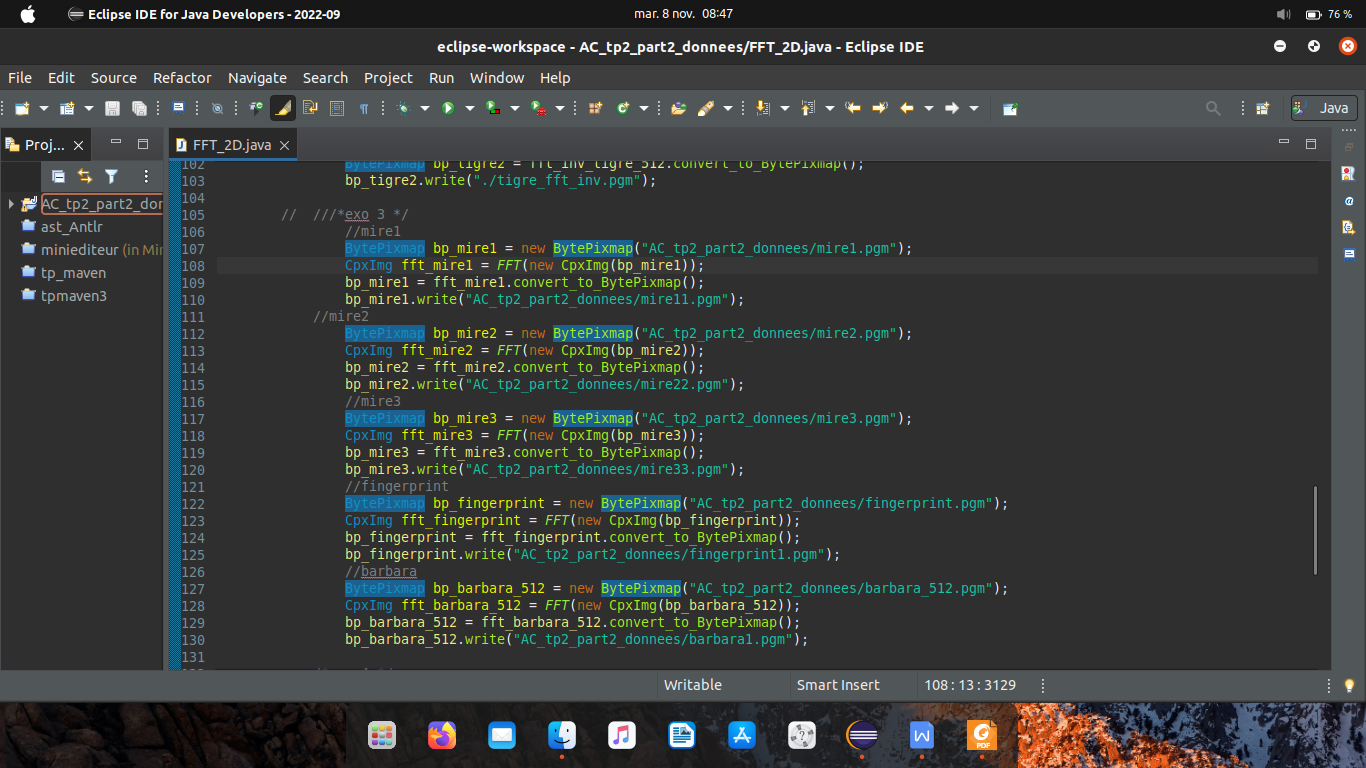
En regardant la photo de mire1.pgm, nous pensons que sa TFD sera sous la forme de points alignés horizontalement, vue que l’image mire.pgm correspond à un ensemble de bande noire et blanche alignées verticalement et aussi en nous référant à la documentation du TP.

Vérifions avec le code ci-dessous :

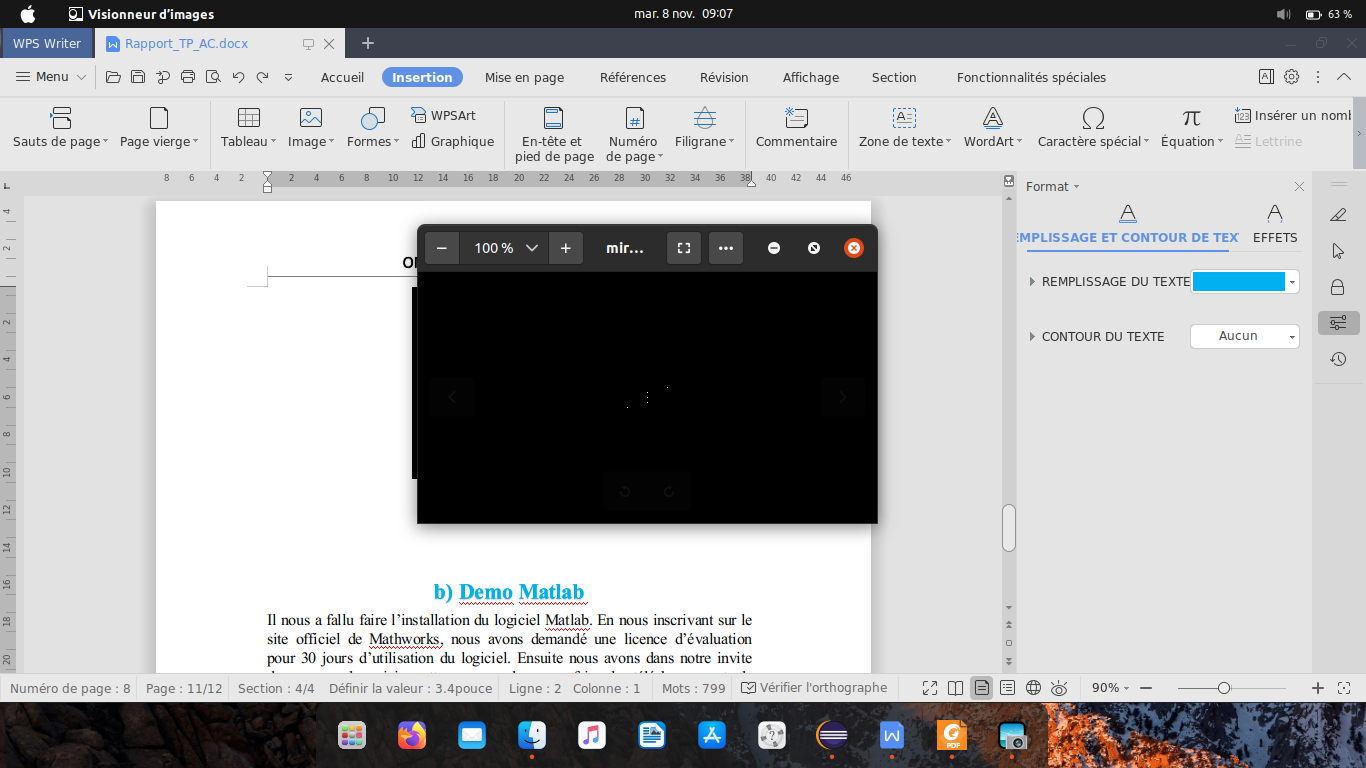
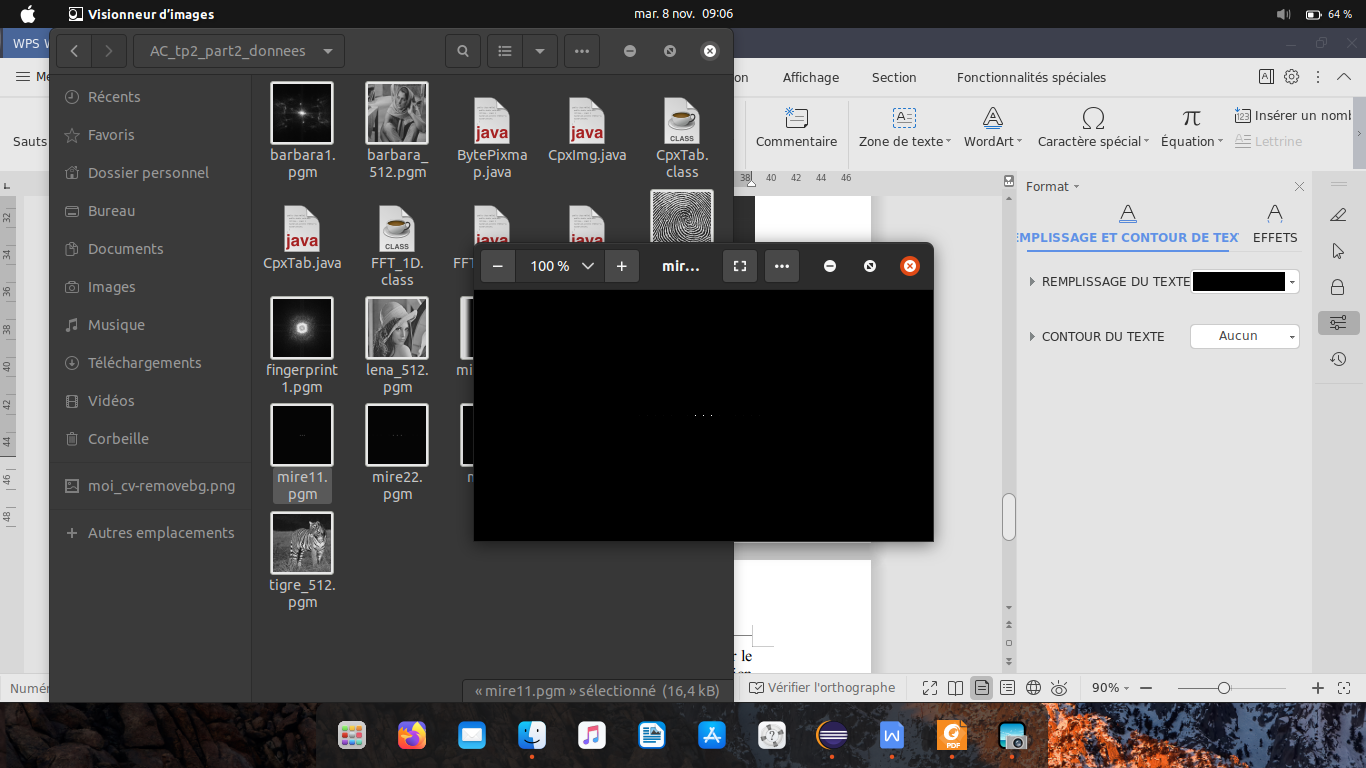
 Le résultat obtenu après exécution est le suivant :



Appliquons la même démarche sur les images de mire2, mire3, fingerprint et l’image de babara, en exécutant le code suivant :

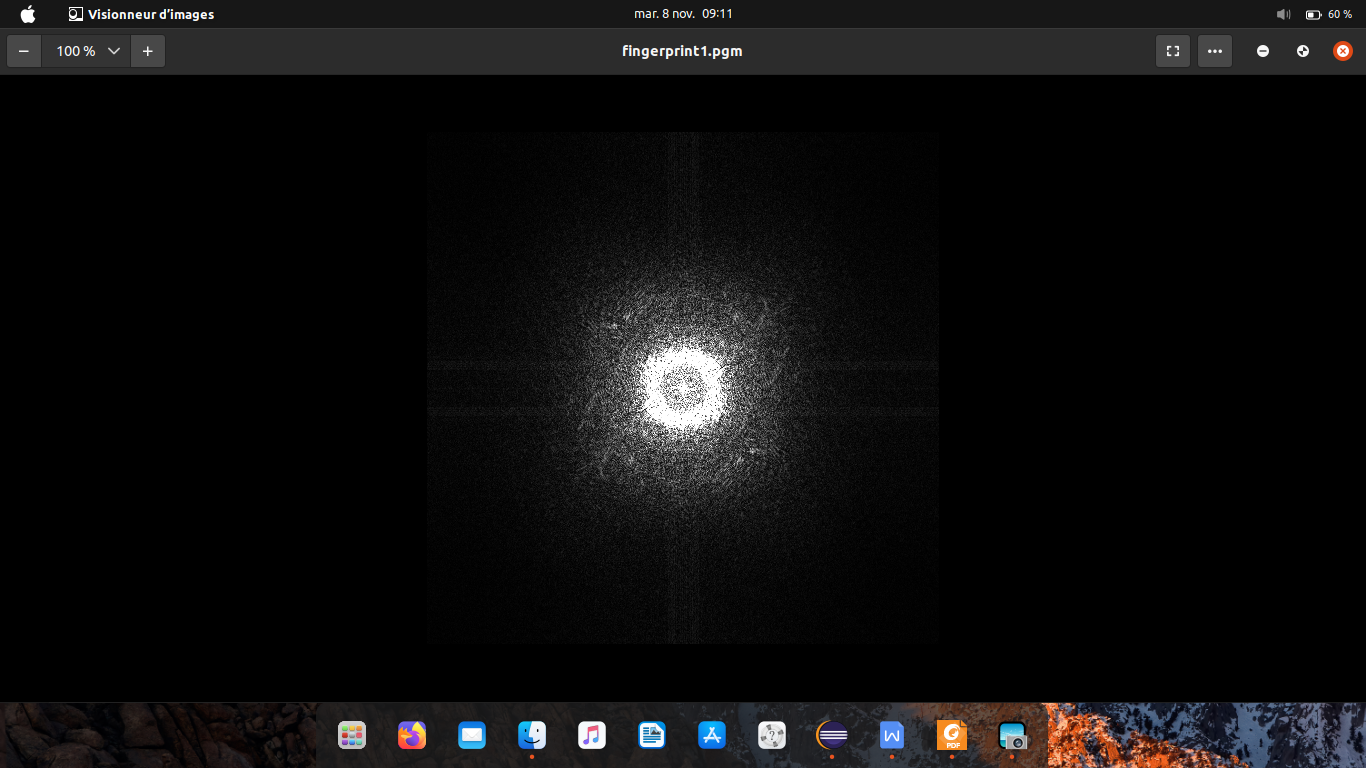


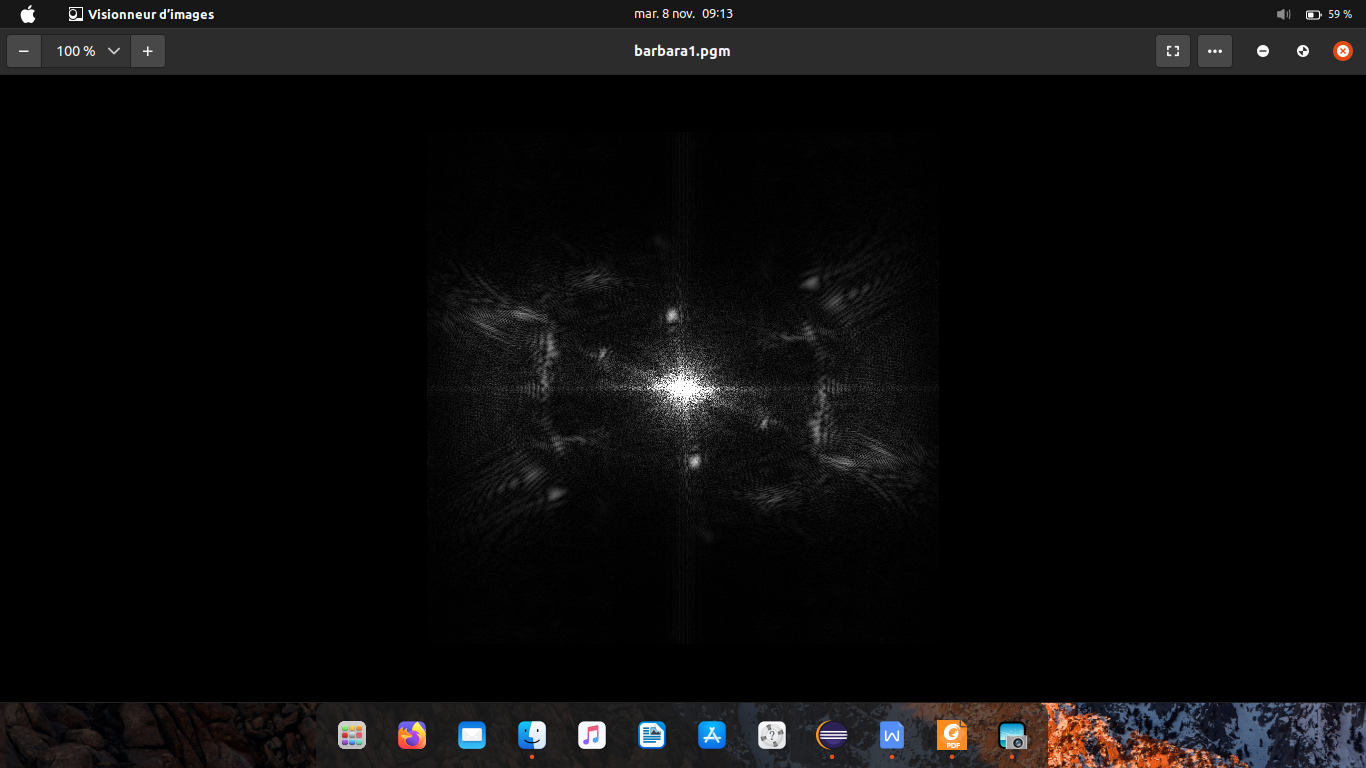
Les résultats obtenus sont les suivants :



**Image :** TFD sur mire 2.pgm

**Image :** TFD sur mire 3.pgm

****

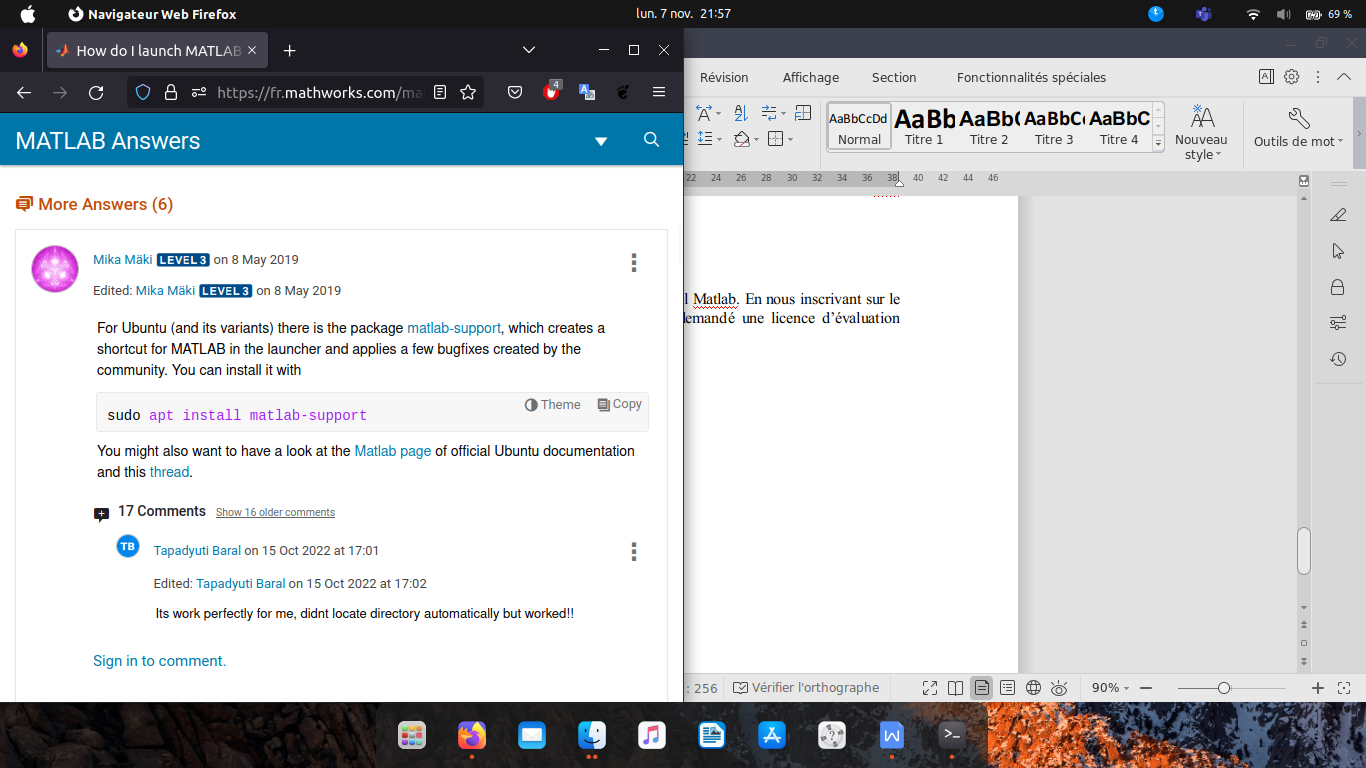
****

**Image :** TFD sur fingerprint.pgm

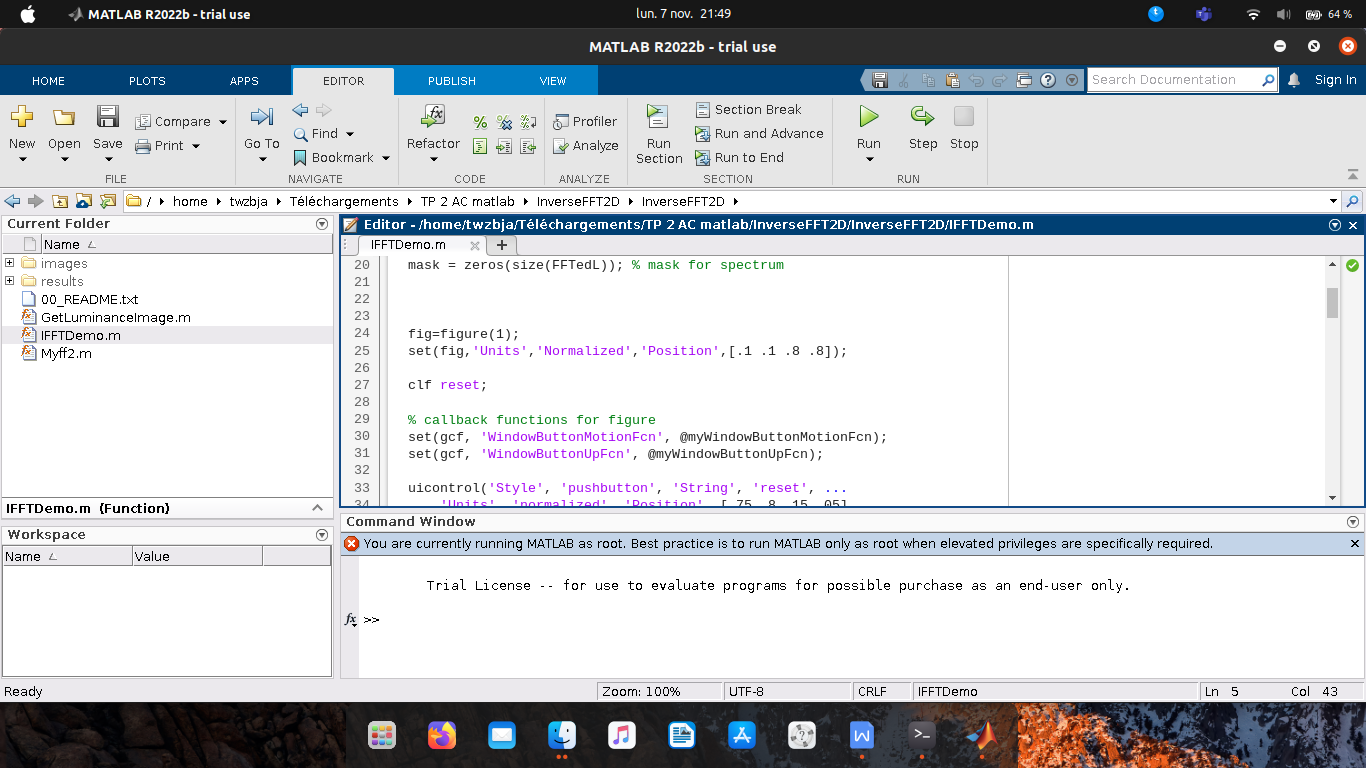
**Image :** TFD sur babara.pgm

1. **Exercice 4**
2. **Demo Matlab**

Il nous a fallu faire l’installation du logiciel Matlab. En nous inscrivant sur le site officiel de Mathworks, nous avons demandé une licence d’évaluation pour 30 jours d’utilisation du logiciel. Ensuite nous avons dans notre invite de commande saisie cette commande pour faire le téléchargement de Matlab :



Après installation du logiciel sur notre machine, nous avons télécharger le dossier « InverseFFT2D ». En tentant d’exécuter le fichier «IFFTDemo.m» sous Matlab. Nous avons constaté une incompatibilité de version, vu que la version du logiciel que nous avons installée est R2022b. Nous avons du faire une adaptation du code, à la version du logiciel Matlab que nous utilisons.



En exécutant notre le code, et lorsque nous cliquons dans la partie FFT pour l’intégration des fréquentielles 2D nous obtenons le résultat suivant :

