FACULTE DES SCIENCES BEN M'SICK
UNIVERSITÉ HASSAN II DE CASABLANCA

PROJET DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME DE MASTER EN MATHEMATIQUES

Présenté par :

Siham Hachoum

Traitement d'images par la transformée de Fourier

Soutenu le 06/06/2022, devant la commission d'examen :

Pr.M. IZID	Faculté des Sciences Ben M'sik	Présidente
Pr. A. ABTA	Faculté Polydisciplinaire, Safi	Examinateur
Pr.Y. EL FOUTAYENI	Faculté des Sciences Ben M'sik	Examinateur
Pr.H. LAARABI	Faculté des Sciences Ben M'sik	Examinateur
Pr.H FERJOUCHIA	Faculté des Sciences Ben M'sik	Examinatrice
Pr.I.AGMOUR	Faculté des Sciences Ben M'sik	Examinatrice
Pr.M.El OURRACHI	Faculté des Sciences Ben M'sik	Examinateur
Pr.O. ZAKARY	Faculté des Sciences Ben M'sik	Co-encadrant
Pr.K. ADNAOUI	Faculté des Sciences Ben M'sik	Encadrant

Année Universitaire 2021-2022

À mes parents :
Mon père qui m'a appris la discipline
Ma mère qui m'a appris la patience
A mes chers frères et mes amis.

Remerciements

Je tiens tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce travail.

J'exprime ma profonde gratitude et respectueuse à mon encadrant Monsieur **Khalid Adnaoui**. Je le remercie de m'avoir encadrée, orientée, aidée et conseillée. J'ai pris un grand plaisir de travailler avec lui.

Mes plus vifs remerciements vont aussi à Pr.O. ZAKARY.

Je remercie également Pr.M. IZID, de l'honneur qu'elle m'a fait en acceptant de présider le jury de ce mémoire.

Je tiens à remercier aussi les membres de jury : Pr. A. ABTA, Pr.Y. EL FOUTAYENI, Pr.H. LAARABI, Pr.H FERJOUCHIA, Pr.I.AGMOUR et Pr.M.El OURRACHI, pour avoir accepté d'évaluer ce travail.

Je profite à cette occasion pour remercier tous mes enseignants et tous les enseignants sans oublier de la faculté des Sciences Ben M'sik, dès le premier cours que j'ai eu lorsque le dernier cours.

Merci !

Table des matières

Introduction

Le traitement d'images désigne une discipline de l'informatique et des mathématiques appliquées s'inscrit dans un processus préliminaire destiné à préparer automatiquement les images à leur analyse, leur interprétation, leur transformation ou leur transmission et ça se fait de plus en plus pressant à mesure que l'image numérique s'impose comme un support et une source d'information privilégiée.
Le traitement d'images est un sous-ensemble du traitement du signal dédié aux images de toutes sortes, tout en opérant dans le domaine numérique.

Il existe différents domaines d'application du traitement d'images :
• En *médecine*, le traitement des images radiologiques, scintigraphiques, scanner permet d'apporter une aide appréciable au diagnostic.
• En *agriculture*, le traitement des images satellitaires (ou télédétection) permet de collecter des informations.
• Dans le *traitement automatique de documents*, la lecture de texte permet la saisie et l'archivage automatique.
• Etc.

Parmi les multiples finalités du traitement d'images, nous nous intéresserons principalement dans ce mémoire aux images numériques et leurs transformations par transformée de Fourier.

La transformée de Fourier est un outil permettant la compréhension et la mise en œuvre de nombreuses techniques numériques de traitement des images. Cet outil trouve de nombreuses applications dans des domaines tels que l'amélioration de la qualité des images, les transmissions numériques, le milieu biomédical, ou encore l'astronomie.
La transformée de Fourier est un outil mathématique essentiel en traitement des images pour deux raisons :

◇ La plupart des dégradations rencontrées lors de la création de l'image s'expriment simplement en termes de transformée de Fourier. Cette dernière permet donc de comprendre le comportement d'une chaîne image.

◇ C'est un exemple historique et important de traitement d'une image à partir de la représentation de l'image dans une base. C'est l'un des grands domaines de recherche actuels.

L'objectif de ce mémoire est de transformée une image numérique dans l'espace de Fourier. On veut savoir comment est-elle à l'entrée ? Quel est l'effet de la transformée de Fourier ? Et que se passe-t-il à la sortie ?. À vrai dire, nous voulons savoir si les informations contenues dans l'image initiale peuvent être conserve après la transformation et la transformation inverse de Fourier.

Trois chapitre sont alors présentés pour à ferme cette étude.

Le premier chapitre a pour objectif de donner les définitions et les concepts de base des images numériques, leurs structures et leurs caractéristiques ainsi que les opérations effectuées sur ces dernières.

Le deuxième chapitre est consacré à la transformée de Fourier des images, nous présenterons un rappel sur la transformée de Fourier continue. Ensuite, nous introduirons la transformée de Fourier discrète. Ainsi, l'implémentation de la transformée de Fourier rapide (FFT) sur les images.

Le troisième chapitre sera entièrement dédié à l'application des principaux algorithmes naïfs et rapides de transformée de Fourier sous MATLAB.

Nous terminerons ce mémoire par une conclusion sanctionne le travail réalisé.

Historique

L'étude du traitement d'images a commencé dans les années 1920 avec la transmission d'images par le câble sous-marin allant de New York à Londres.
Harry G. Bartholomew et **Maynard D. McFarlane** ont effectué la première numérisation d'image avec compression de données pour envoyer des fax de Londres à New York. Le temps de transfert passe ainsi de plus d'une semaine à moins de trois heures. Il n'y avait pas d'évolution par la suite jusqu'à la période d'après-guerre.

Le traitement du signal a évolué vers la fin de la Seconde Guerre mondiale avec l'arrivée du radar. La prospection pétrolière a participé aussi beaucoup au développement des techniques de traitement du signal.

Le véritable essor du traitement d'images n'avait lieu que dans les années 1960, quand les ordinateurs ont commencé à être suffisamment puissants pour travailler sur les images. Peu après, la redécouverte de la transformée de Fourier rapide (FFT) révolutionne le domaine, en rendant possible les manipulations du contenu fréquentiel des signaux sur l'ordinateur. Cependant, l'essentiel des recherches porte encore, à cette époque, sur l'amélioration des images et leur compression.
En 1980, **David Marr** a été le premier qui a formulé la détection de contours d'une manière précise [1]. Au cours des années 1980, un véritable engouement se fait jour pour le traitement de l'image et surtout pour la compréhension de l'image par des systèmes experts.

Les années 1990 sont témoins de l'amélioration constante des opérateurs. La recherche médicale devient un grand demandeur en traitement d'images pour améliorer les diagnostics faits à partir des nombreuses techniques d'imagerie médicale, la technique reine étant l'IRM [î]. Les publicitaires, puis le grand public se familiarisent avec la retouche d'image grâce au logiciel Photoshop et au traitement d'images dans un objectif esthétique.

Chapitre 1

Les images numériques

1.1 Qu'est-ce qu'une image ?

Cette simple question admet en réalité plusieurs réponses, plus ou moins techniques, selon le domaine étudié. Commençons par donner la réponse qui vient naturellement à la majorité d'entre nous : une image est une représentation visuelle de quelque chose ou de quelqu'un.
En mathématiques, une image est une fonction. Cette fonction quantifie l'intensité lumineuse de n'importe quel point dans l'image.

N.B : Une image est un signal 2D S(x,y).

1.2 Image numérique

Une image est dite numérique lorsque sa sauvegarde est obtenue sous forme binaire (suite de 0 et de 1). Donc l'image numérique fait appel à l'informatique. Chaque image numérique est constituée d'un nombre donné de lignes et de colonnes. Chaque ligne chaque colonne comporte un nombre de point donnés. L'ensemble de ces points constitue une matrice. Ces points sont dénommés pixel. Chaque « case » de cette matrice contient des nombres caractéristiques à la couleur attribuée au pixel. **(Une image = 0,5 à 6 millions de pixels)**.

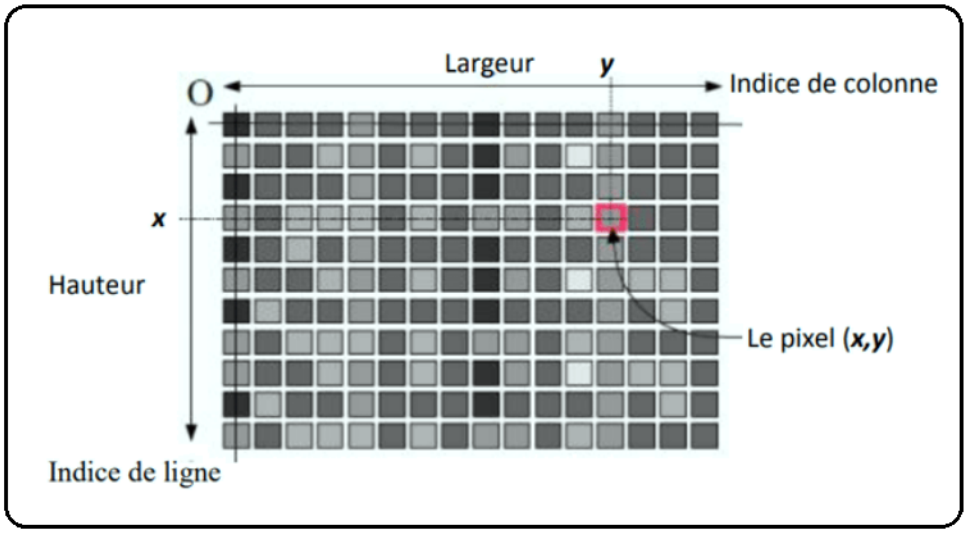


FIGURE 1.1 – Représentation d'une image numérique

1.3 Acquisition des images numériques

Le processus général pour l'acquisition d'une image numérique est illustré sur la figure suivante :

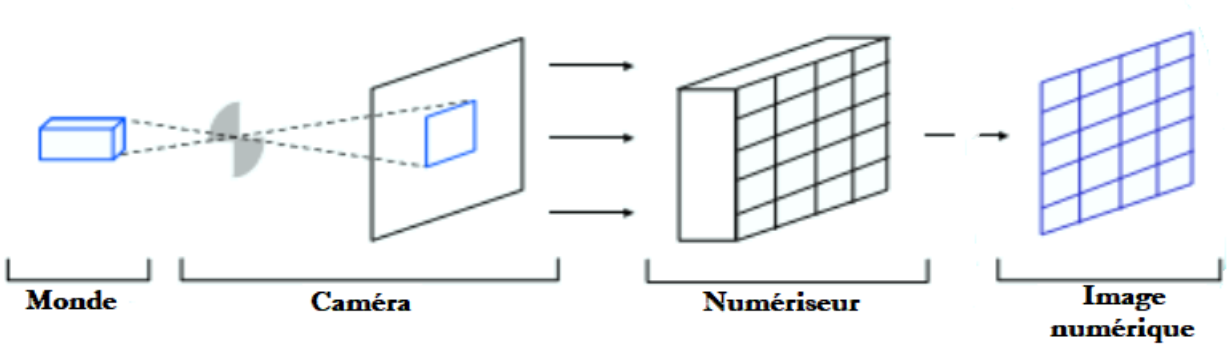


FIGURE 1.2 – l'acquisition d'image

Deux procédés sont impliqués pour numériser une image,

Numérisation = Échantillonnage + Quantification

• Echantillonnage

L'échantillonnage est le procédé de discrétisation spatiale d'une image consistant à associer à chaque pixel une unique valeur.

• Quantification

La quantification désigne la discrétisation tonale correspondant à la limitation du nombre de valeurs différentes que peut prendre chaque pixel.

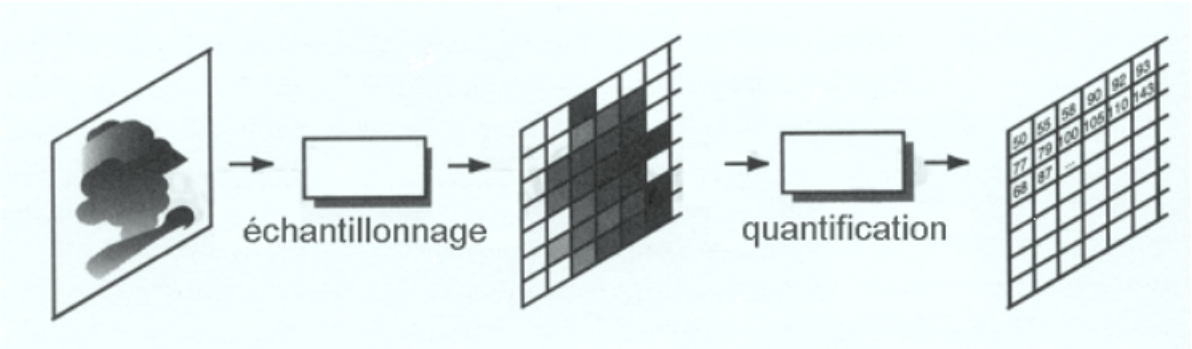


FIGURE 1.3 – Echantillonnage et quantification

1.4 Les types d'images numériques

D'un point de vue formel, les images numériques peuvent se présenter sous deux aspects, images vectorielles et images matricielles :

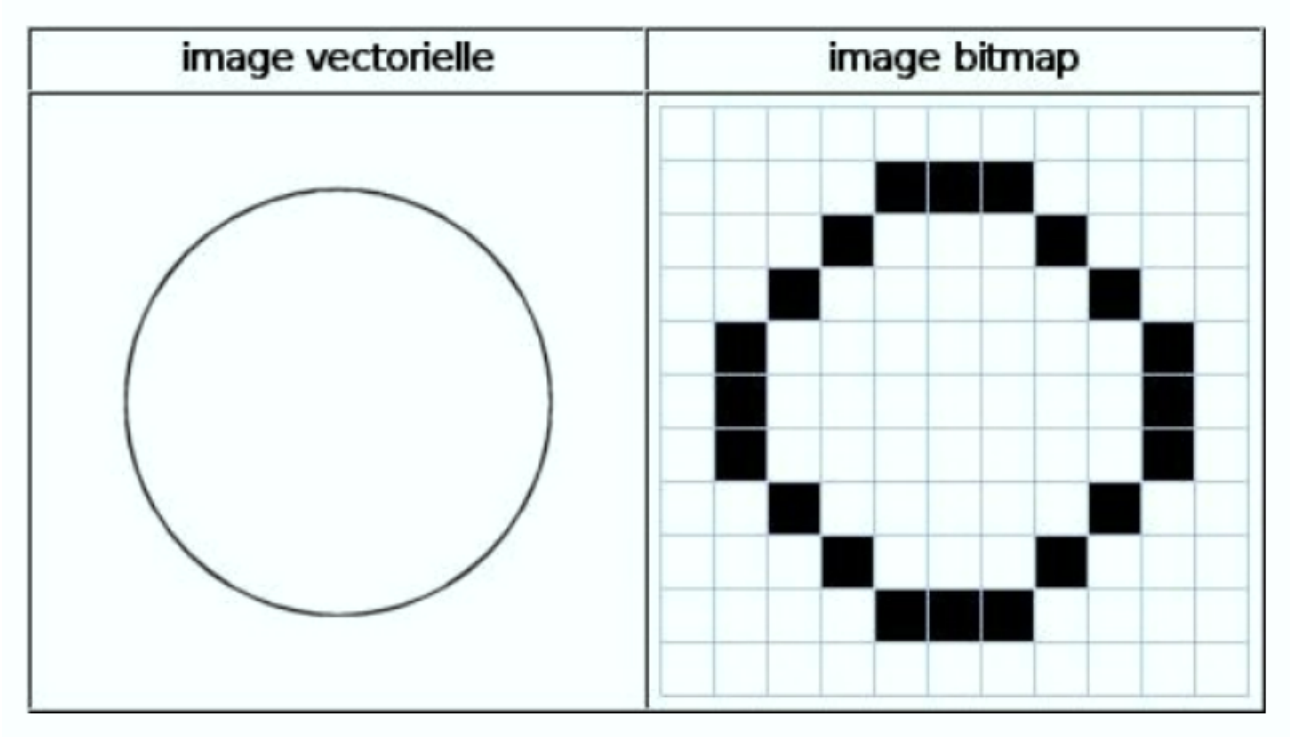


FIGURE 1.4 – Les types des images numériques

1.4.1 Images matricielles (Bitmap)

L'image matricielle ou l'image en mode point est utilisée pour les captations (photo, vidéo, numérisation de documents), mais aussi en illustration pour la création de dessins, à l'aide d'une palette graphique. C'est encore une méthode de dessin numérique très utilisée par les illustrateurs, car les techniques utilisées dans les logiciels comme Photoshop se rapprochent des méthodes traditionnelles. Mais le dessin vectoriel fait de plus en plus d'adeptes.

Une image matricielle est composée de petits carrés d'une seule couleur chacun, appelés Pixels, rangés en ligne et en colonne, sous forme de tableau.
Le pixel est l'élément de base d'une image matricielle.

1.4.2 Images vectorielles

Les images vectorielles sont composées de formes géométriques et sont décrites sous une forme mathématique.
Les images vectorielles sont beaucoup plus économiques en poids, car l'image est définie par un certain nombre de tracés simples (et non en pixels), tels que segments de droites, arcs de cercles ou courbes.
On peut définir simplement ces courbes à partir de quelques paramètres, (par exemple les coordonnées de son centre et la longueur de son rayon définissent un cercle). Chaque objet peut être modifié, déplacé, agrandi, dupliqué... à volonté.
Ce type d'image se prête parfaitement à la création de dessins et illustrations, soit à l'aide de la souris, soit d'une tablette graphique.

Les logotypes et les pictogrammes sont très souvent réalisés sous forme vectorielle, ce qui autorise leur utilisation dans tous les formats, sans aucune perte de qualité.

1.4.3 Avantages et inconvénients des images matricielles et vectorielles

L'avantage des images matricielles est que l'utilisation de pixel est parfaitement adaptée aux images travaillées avec des effets, des ombres, des dégradés ce qui est beaucoup plus difficile à rendre avec des vecteurs. Mais ceci implique quelques inconvénients. Tout d'abord, une image matricielle est créée pour être mise à une certaine taille qui doit être globalement fixe, donc lorsqu'on zoom sur une image bitmap ou qu'on l'imprime, on peut voir une perte de qualité plus ou moins importante qui se traduit par l'apparition de pixels.

Les images vectorielles possèdent comme avantages l'inverse des images matricielles ! C'est-à-dire qu'elles ont la possibilité de l'agrandir indéfiniment sans perdre de qualité initiale ou se déformer, indépendamment de sa taille et de sa résolution.
Par contre, elles sont un peu trop "rigides" au niveau des formes, ce qui fait qu'elles se prêtent surtout à l'illustration et au graphisme.
Elle reste limitée en termes de réalisme. Chaque forme d'une image vectorielle est remplie d'une seule couleur dite solide ou d'un dégradé de couleurs.
Une image vectorielle ne peut pas être affichée directement sur un écran. Elle doit auparavant être transformée en image de type bitmap.

N.B : On peut facilement transformer une image vectorielle en image matricielle par contre le contraire est bien plus complexe.

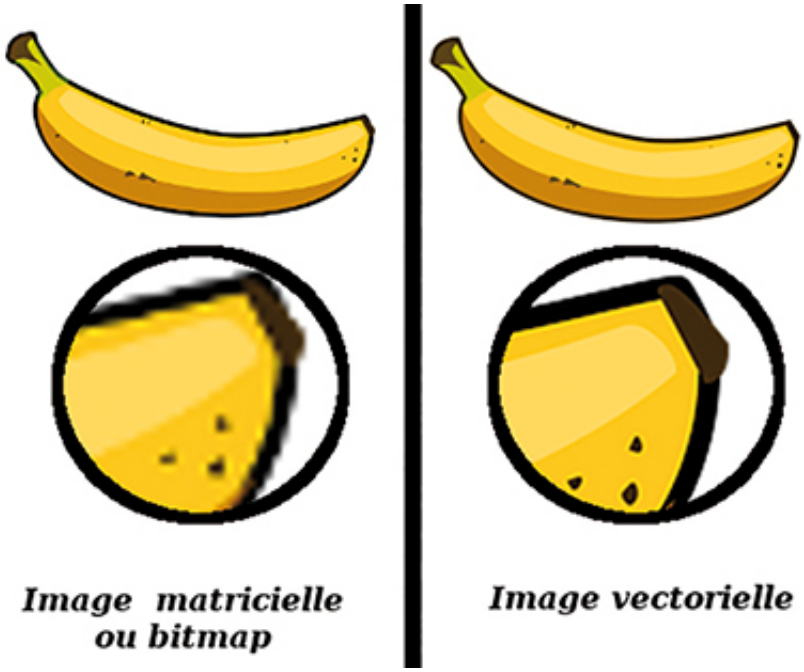


FIGURE 1.5 – Zoom sur image matricielle et image vectorielle

1.5 Les caractéristiques d'une image numérique

1.5.1 Pixel

Un pixel est tout simplement un point. Lorsqu'on agrandit une image numérique, on voit que celle-ci est composée d'un ensemble de "points", appelés pixels.
Le pixel (abréviation venant de l'anglais : picture element) est l'élément de base d'une image ou d'un écran. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions (largeur et hauteur) constituant l'image.
Un pixel est le plus petit constituant d'une image, un point élémentaire caractérisé par une couleur.

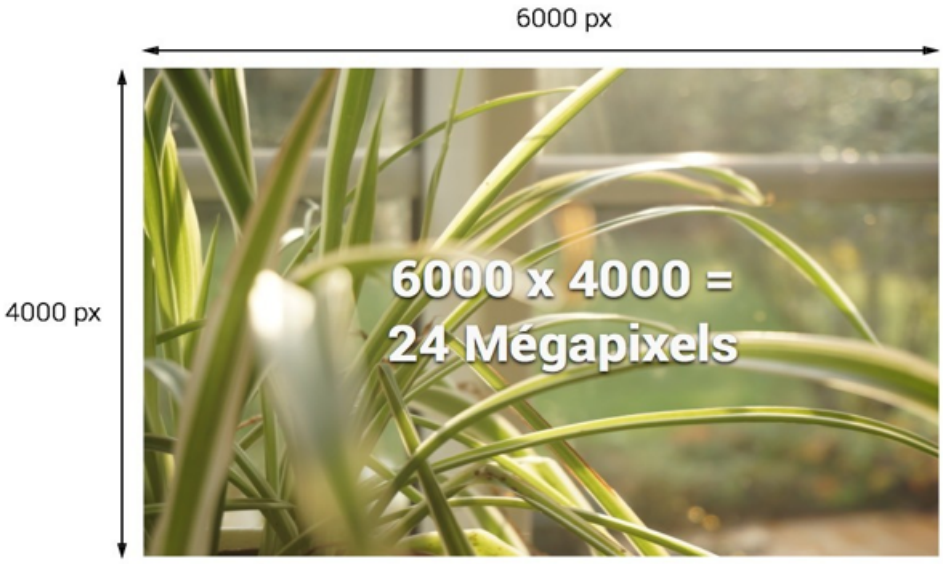
1.5.2 Définition (pixellisation)

La définition d'une image, appelée aussi pixellisation, est le nombre de pixels composant une image, c'est-à-dire sa « dimension informatique ». Plus l'image possède de pixels plus la " définition " de l'image est importante. Le nombre de pixels est directement lié à la qualité du capteur et de l'appareil photographique. L'image en très haute définition donnera davantage une image riche en détail et favorisera la possibilité d'agrandissement de l'image.

Pour calculer la définition d'une image numérique, il nous suffit de multiplier le nombre de pixels sur la hauteur par le nombre de pixels sur la largeur de l'image.

Exemple :

Une image de 6000 x 4000 px a une définition de 24 millions de pixels, ou 24 mégapixels.



N.B : La définition de l'image ne permet pas de connaître ses dimensions, nous savons juste combien de pixels elle est composée !

Définition en pixels = (taille en cm x dpi) / 2.54

1.5.3 Résolution

La résolution d'une image s'exprime en nombre de pixels par unité de mesure. C'est une notion de densité de pixels, et cette résolution ne devient intéressante que lorsque l'on commence à imprimer ses images.
La résolution de l'image fait le lien entre l'unité informatique, qu'est la définition de l'image, exprimée en pixels, et la dimension physique, qui est la taille de l'image par exemple les centimètres d'un tirage papier. La résolution correspond donc à la densité de pixel d'un tirage photo. De manière conventionnelle, on parle de dpi (dots per inch, ou pixels par pouce).
Cette résolution permet d'adapter son image en fonction de son utilisation.

- **Rappel :** un pouce (ou inch) mesure 2,54 cm.

Les résolutions les plus fréquentes sont :
▷ Pour le web : résolution de 72 dpi (basse résolution).
▷ Pour un tirage photo professionnel : 300 dpi (haute résolution).

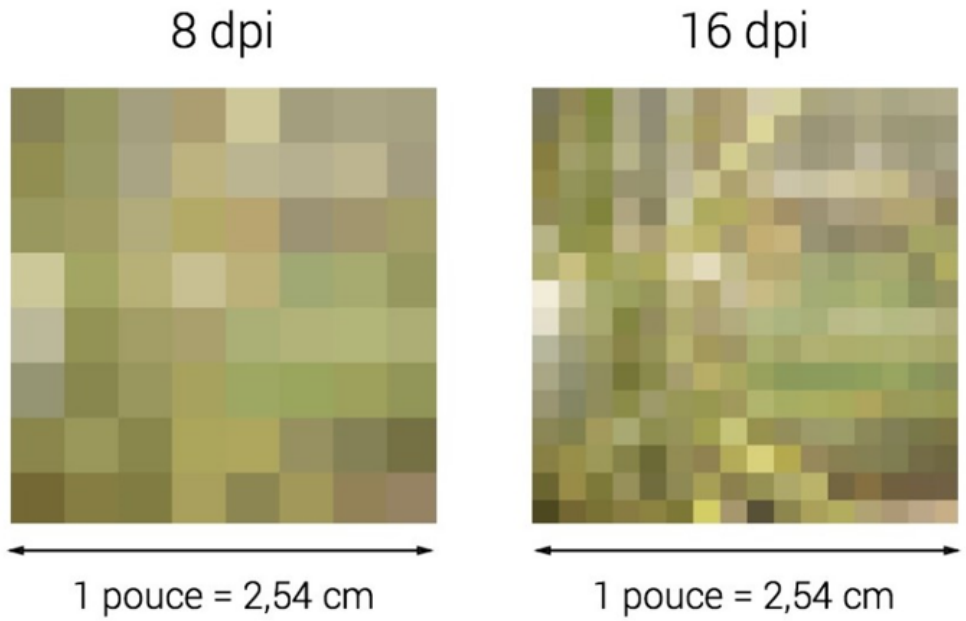


FIGURE 1.6 – Exemple de résolution d'une image

—∘ **Plus la valeur de résolution est élevée** :

- Moins l'image imprimée sera grande (les pixels sont plus « serrés »)
- Plus les détails seront théoriquement fins

—∘ **Moins la résolution est élevée :**

- Plus l'image imprimée sera grande (les pixels sont moins « serrés »)
- Moins les détails seront fins.

1.6 Les formats d'image

Un format d'image est une représentation informatique de l'image, associée à des informations sur la façon dont l'image est codée et fournissant éventuellement des indications sur la manière de la décoder et de la manipuler. Il existe des différents types formats de fichiers images, chaque format d'image est optimisé pour une utilisation différente.

• Formats d'images matricielles

Les formats matriciels les plus utilisés sont :

Format	Signification	Usage
BMP	BitMaP	Format Windows par défaut Fichiers lourds. Adapté à l'impression.
TIF	Tagged Image File	Format propriétaire. Fichiers lourds. Adapté à l'impression.
GIF	Graphics Interchange Format	Schémas, images animées. Permet la transparence d'image.
PNG	Portable Network Graphics	Logos, icones. Photographies.
JPG	Joint Photographic Experts Group	Photographies. Les images de grandes tailles.

• Formats d'images vectorielles

Les formats vectoriels les plus utilisés sont :

Format	Signification	Usage
SVG	Scalable Vector Graphics (graphique vectoriel extensible)	Cartes, schémas, graphiques. Images synthétiques.
SWF	ShockWave Flash Animation Flash	Animations Flash. Applications multimédias.
DWG	DraWinG	DAO. Fichiers AUTOCAD.

1.7 La colorisation d'une image numérique

Une image est définie par l'étendue des teintes de gris ou des couleurs que peut prendre chaque pixel.
Pour une image numérique on distingue 3 grands types de couleurs :

- **Niveaux de gris**
- **Noir et blanc**
- **Couleurs**

1.7.1 Image en niveaux de gris (Monochromie)

Ce qu'on appelle image noir et blanc dans le langage courant est appelé image en niveaux de gris en imagerie numérique.
Une image en niveaux de gris est une image faite avec un dégradé de gris entre le noir et le blanc. En général, les images en niveaux de gris renferment 256 teintes de gris. Par convention la valeur zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (intensité lumineuse maximale).

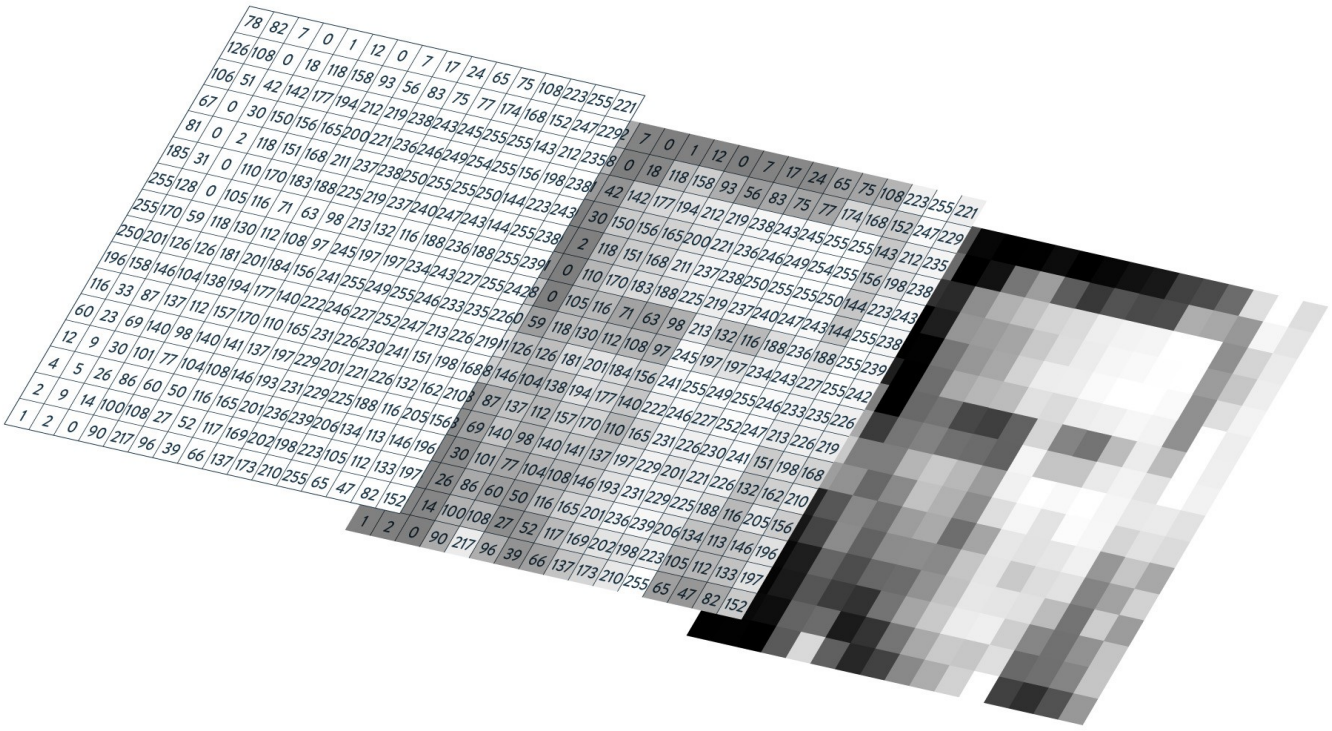


FIGURE 1.7 – Visualisation d'une image en niveaux de gris, ainsi que des valeurs d'intensité correspondantes.

Le nombre 256 vient de la représentation informatique des nombres entiers.
L'unité de base manipulée par les ordinateurs est le bit, dont la valeur vaut soit 0 soit 1.
Ces bits sont le plus souvent regroupés en octets, contenant 8 bits.
Le nombre d'octets différents est égal à

$$2 * 2 * ... * 2 = 2^8 = 256.$$

En quantifiant les niveaux de gris sur un octet, on peut ainsi représenter 256 niveaux de gris différents.
Certains systèmes d'acquisition permettent d'acquérir des images avec une intensité codée sur 12, 14 ou 16 bits, permettant de représenter jusqu'à 4096,16384 ou 65536 valeurs différentes. L'intérêt de ce nombre élevé de niveaux est de pouvoir détecter des variations plus subtiles, mais leur visualisation nécessite des logiciels adaptés.

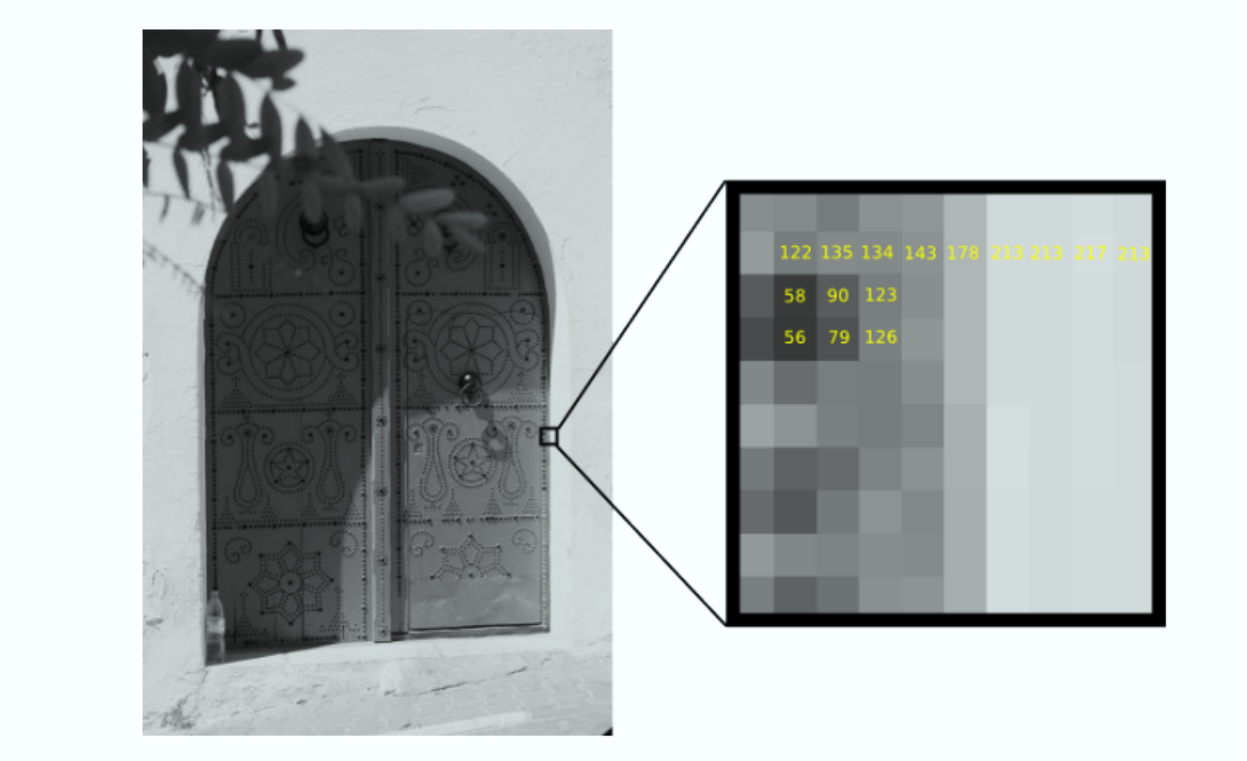


FIGURE 1.8 – Sous image en niveaux de gris

1.7.2 Image noir et blanc (binaire)

Une Image noir et blanc ou image binaire est une image en niveau de gris particulière, qui n'a que deux valeurs possibles où chaque point peut prendre uniquement la valeur 0 ou 1. Les pixels sont noirs (0) ou blancs (1).

Une image noir et blanc est donc ni plus ni moins qu'un « échiquier » non régulier et sa matrice et une matrice de dimension 2 n'ayant que des valeurs 0 et 1.

1.7.3 Image couleur (Trichromie)

Principe

Une image couleur est en réalité la composition de trois images en niveaux de gris sur trois composantes.

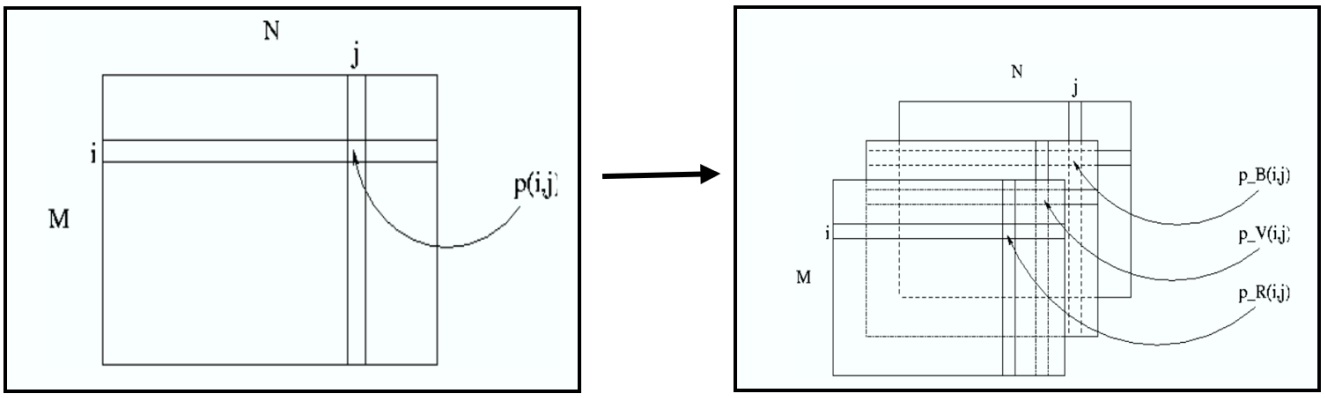


FIGURE 1.9 – La décomposition d'une image couleur

On définit donc trois plans de niveaux de gris, un rouge, un vert et un bleu. Chacune de ces trois images s'appelle un canal.

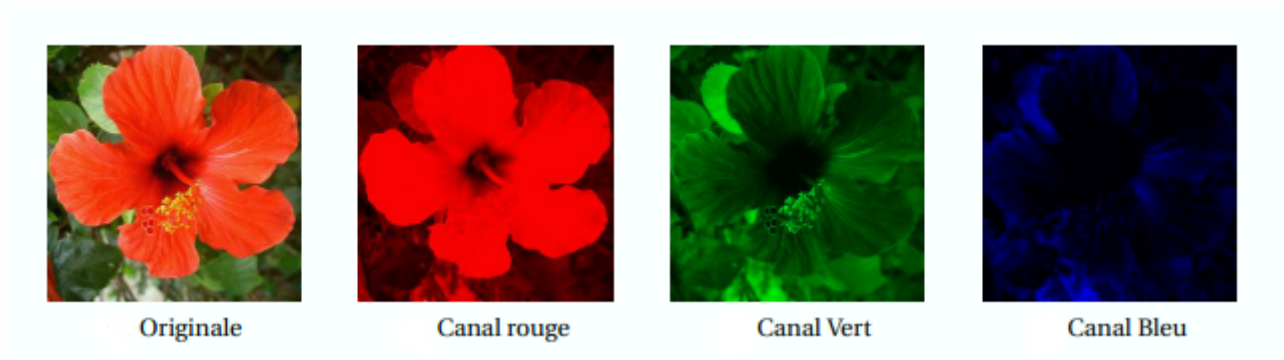


FIGURE 1.10 – Canaux RVB

Chaque pixel de l'image couleur contient ainsi trois nombres (r,v,b), chacun étant un nombre entier entre 0 et 255. Si le pixel est égal à (r,v,b) = (255,0,0), il ne contient que de l'information rouge, et est affiché comme du rouge. De façon similaire, les pixels valant (0,255,0) et (0,0,255) sont respectivement affichés vert et bleu.

La couleur finale est obtenue par synthèse additive des ces trois composantes.

Synthèse des couleurs

La synthèse de la couleur consiste à reproduire l'ensemble des couleurs visibles à partir d'un petit nombre de couleurs, appelées couleurs primaires. Dans la nature, il existe deux principes de combinaison de couleurs primaires :

La synthèse additive : utilisé dans les écrans vidéo ou en projection ;
La synthèse soustractive : on rencontre en peinture ou en impression.

- **Synthèse additive**

Le principe de la synthèse additive des couleurs repose sur l'addition, dans différentes proportions, des trois couleurs primaires, le Rouge(r), le Vert(v) et le Bleu(b), pour reconstituer l'ensemble des couleurs perceptibles par l'oeil. On parle de synthèse additive, car en mélangeant deux des couleurs primaires, on obtient une couleur plus claire que chacune des composantes colorées initiales.
Par exemple, le mélange du rouge et du vert donne la couleur jaune qui est incontestablement plus claire. Si le mélange des deux couleurs primaires se fait dans des proportions identiques, la couleur résultante obtenue est appelée couleur complémentaire de la troisième. Les couleurs cyan, magenta et jaune sont respectivement les complémentaires du rouge, du vert et du bleu. En additionnant dans les mêmes proportions les trois couleurs primaires, on obtient un niveau de gris.

La figure suivante montre les règles de composition cette synthèse additive des couleurs.

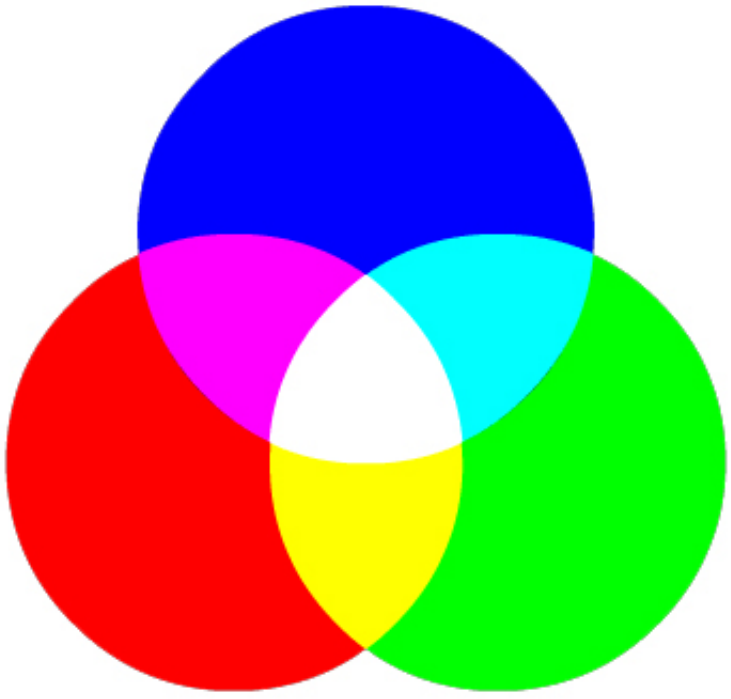


FIGURE 1.11 – Synthèse additive des couleurs

- **Synthèse soustractive**

La synthèse soustractive est Une autre représentation courante pour les images couleurs utilise comme couleurs de base le Cyan (c), le Magenta (m) et le Jaune (j).
On calcule les trois nombres (c,m,j) correspondant à chacun de ces trois canaux à partir des canaux rouge, vert et bleu (r,v,b) comme suit :

$$\mathbf{c = 255 - r} \qquad \mathbf{m = 255 - v} \qquad \mathbf{j = 255 - b}$$

Par exemple, un pixel de bleu pur $(r, v, b) = (0, 0, 255)$ va devenir $(c, m, j) = (255, 255, 0)$.

FIGURE 1.12 – Canaux CMJN

La figure suivante montre les règles de composition cette synthèse soustractive des couleurs.

FIGURE 1.13 – Synthèse soustractive des couleurs

Remarque 1 :
Pour comprendre pourquoi elles s'appellent additive et soustractive, il faut se rappeler qu'en physique, le noir est l'absence de rayons lumineux, alors que le blanc est la présence de l'ensemble du spectre visible, Ainsi, on peut imaginer la synthèse additive comme partir du noir complet et ajouter des sources de lumières colorées, et la synthèse soustractive comme partir de l'ensemble du spectre lumineux (le blanc) et retirer peu à peu certaines couleurs du spectre.

Remarque 2 :
L'œil humain ne peut discerner que 300 000 couleurs différentes et environ une trentaine de niveaux de gris.

1.7.4 Transformer une image couleur en niveaux de gris

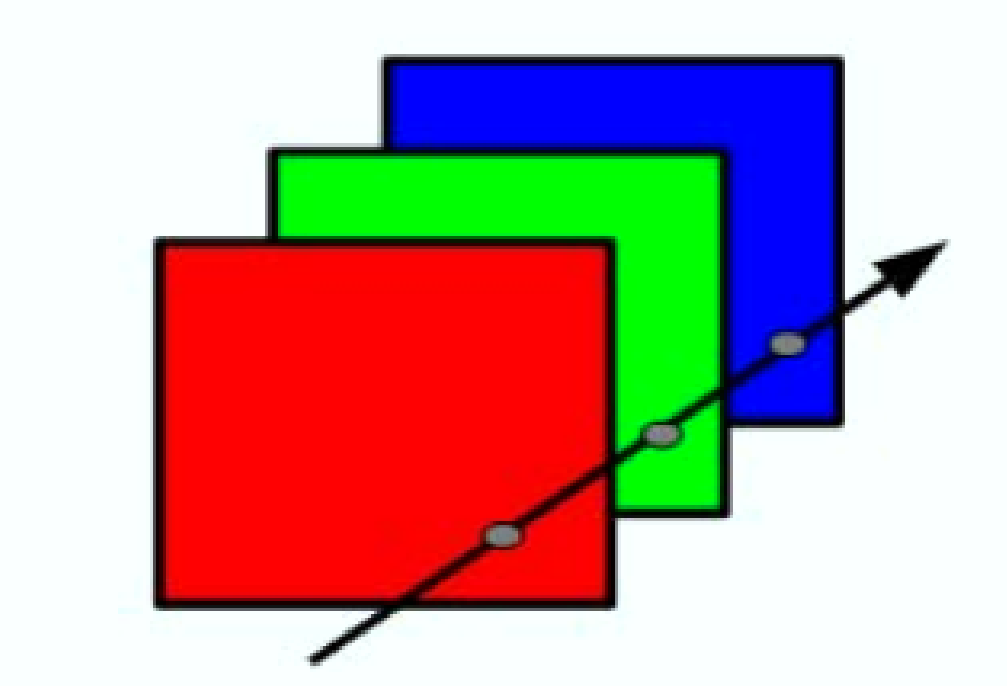
Les notions de traitement d'image sont d'abord définis pour les images en niveaux de gris puis appliqués sur des images en couleurs.

Principe :

On peut calculer une image en niveaux de gris à partir d'une image couleur (RGV) en moyennant les trois canaux (r,v,b). On calcule donc une valeur

$$\mathbf{a = (r + v + b) \ / \ 3}$$

qui s'appelle **la luminance** de la couleur.



La figure suivante montre l'image de luminance associée à une image couleur.

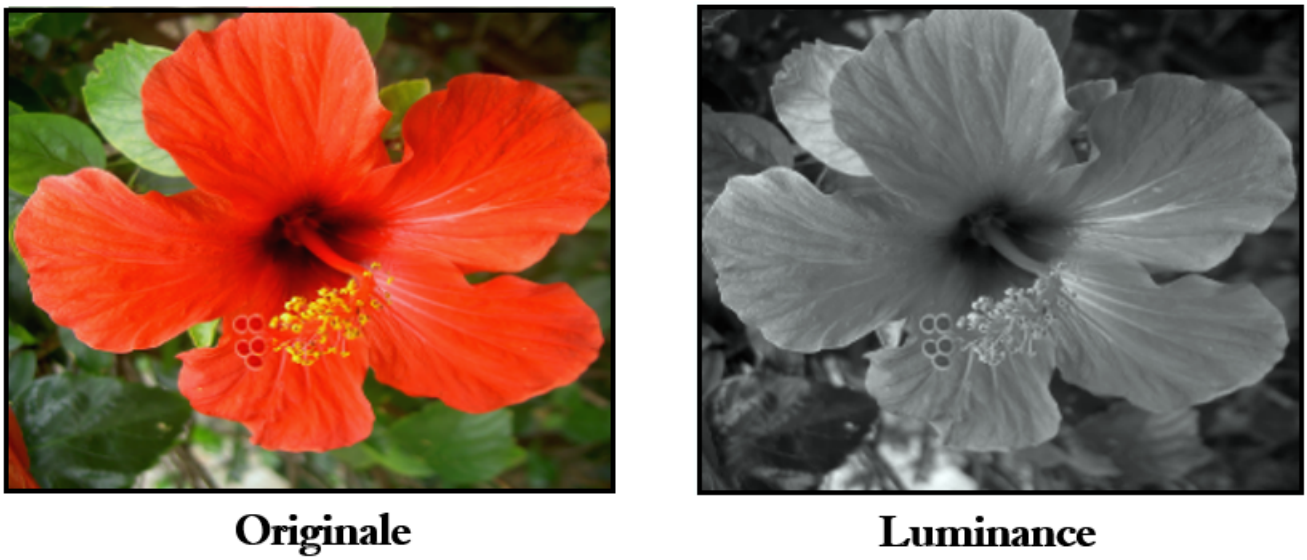
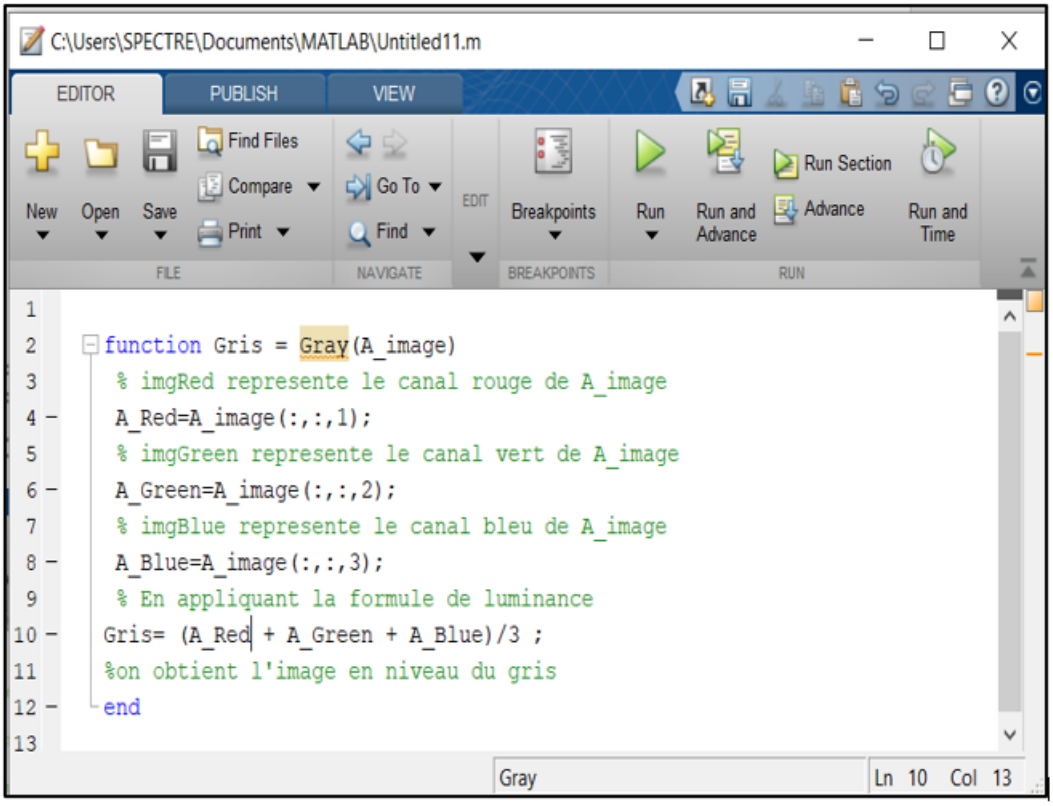


FIGURE 1.14 – Luminance

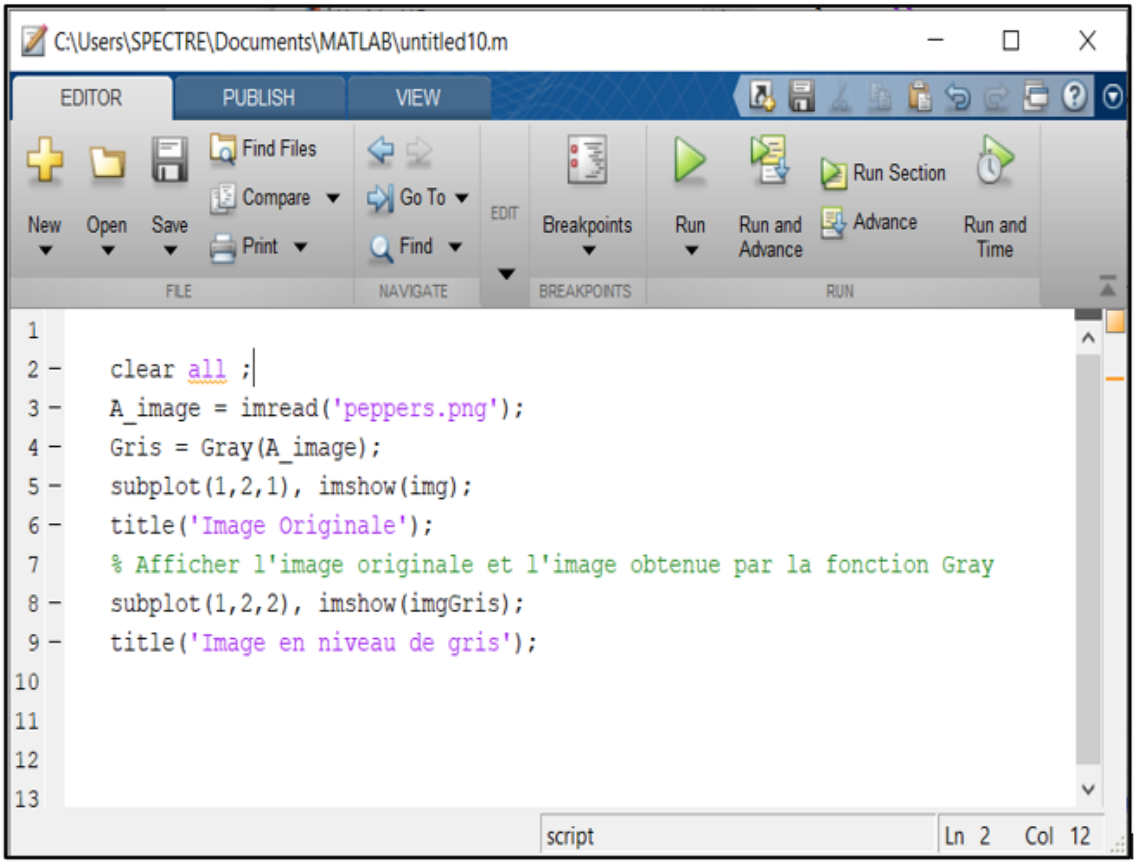
Application sous Matlab

- 1^{er} **méthode : Notre fonction**

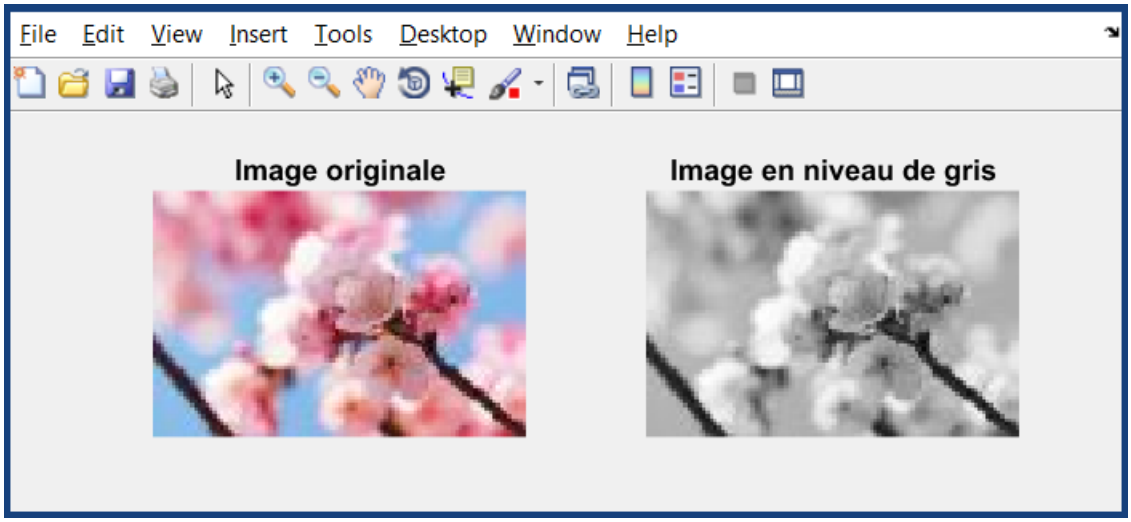
—∘ Fonction :



—◦ Code d'exécution :



—◦ Production :

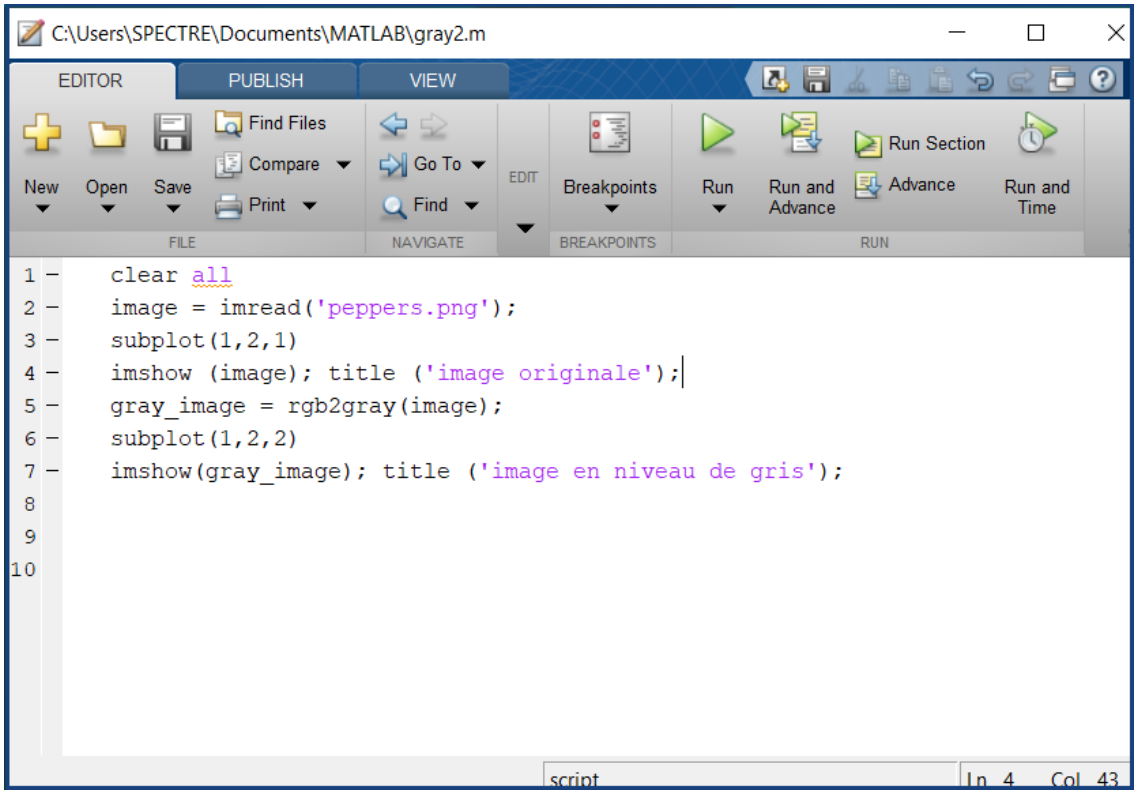


Dans la figure ci-dessus, l'image de gauche est l'image couleur d'entrée et l'image de droite est le résultat de la conversion.

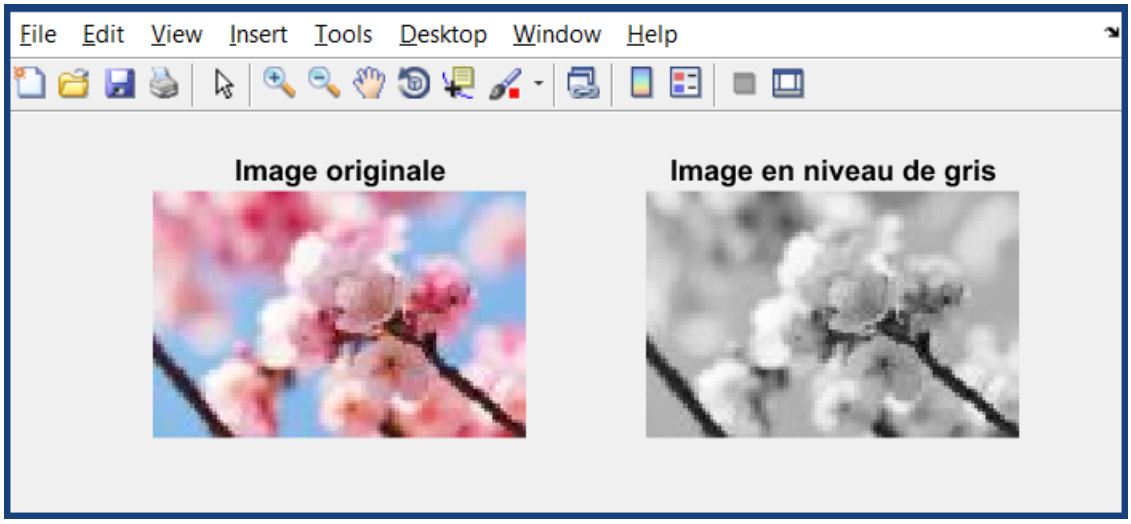
- **2^{éme} méthode : La fonction MATLAB "rgb2gray()"**

Nous pouvons transformer une image couleur en niveaux de gris à l'aide de la fonction **rgb2gray()** dans MATLAB.

—∘ Code :



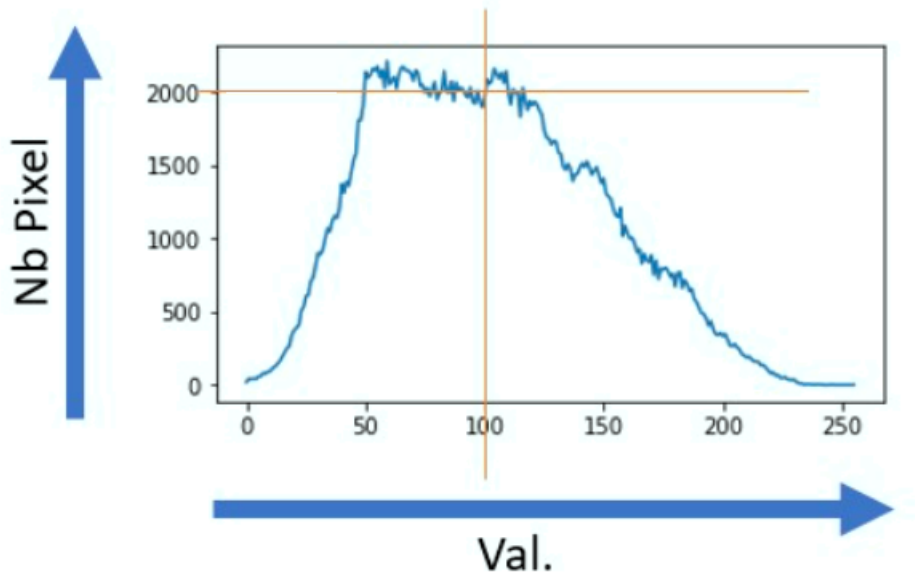
—∘ Production :



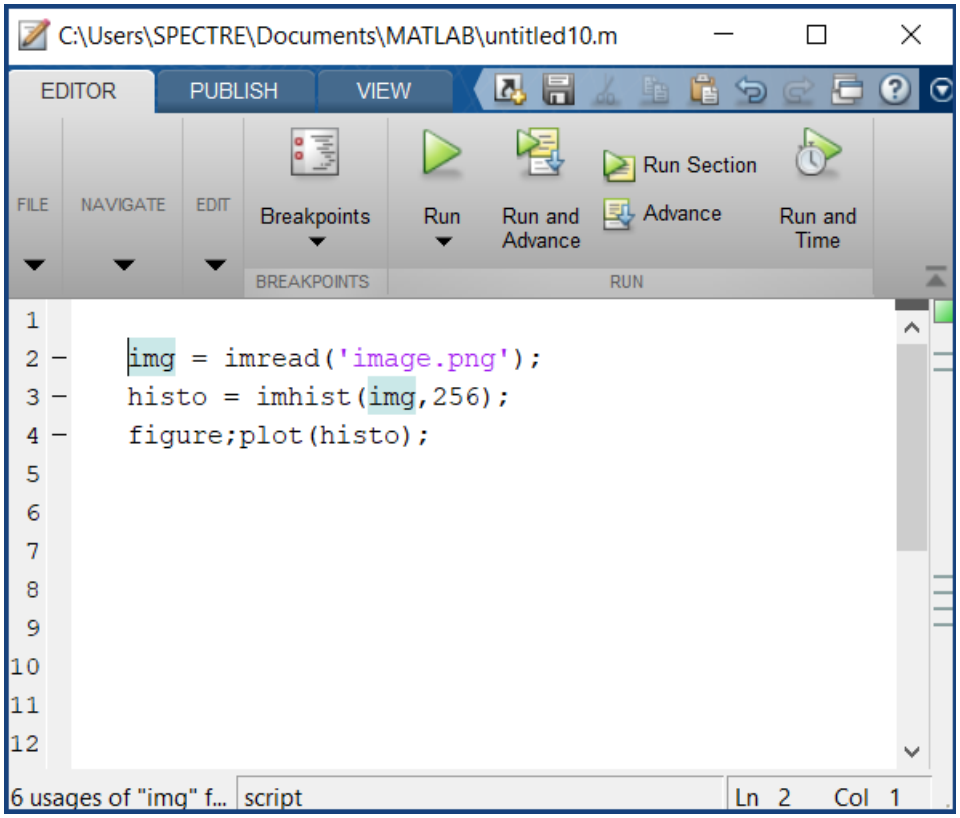
Dans la figure ci-dessus, l'image de gauche est l'image couleur d'entrée et l'image de droite est le résultat de la conversion.

1.8 L'histogramme

L'histogramme en photo, c'est la carte d'identité de notre image. Une courbe statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (noir pur)(à gauche) au plus clair (blanc pur)(à droite), alors que de bas en haut (en ordonnées) sont représentés le nombre de pixels pour une teinte donnée.

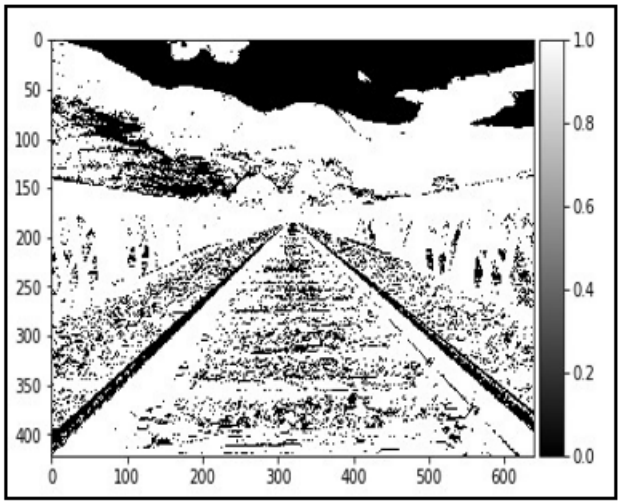


⊸ **Code MATLAB :**(pour les images en niveaux de gris)

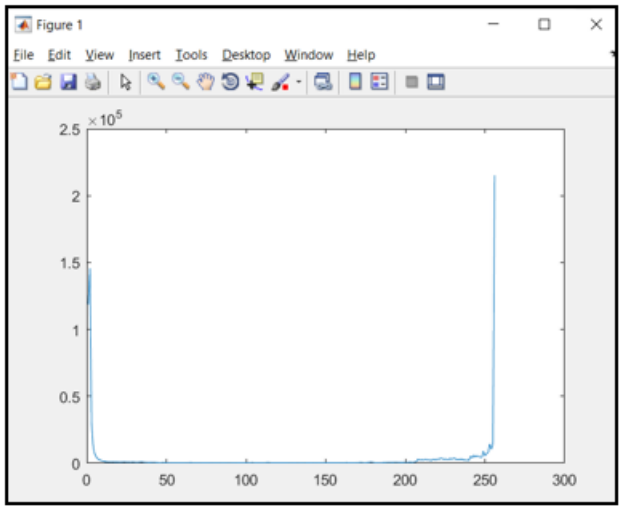


• L'histogramme d'image binaire

Une image en noir et Blanc a un histogramme très basique (binaire) qui ne comporte que des valeurs 0 ou 1(255).



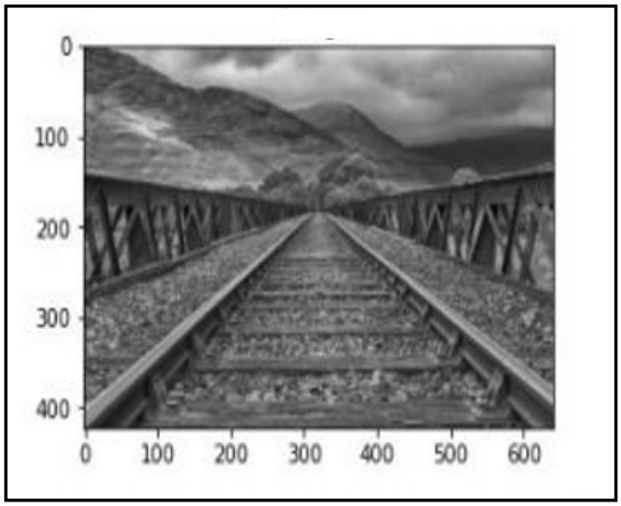
Image



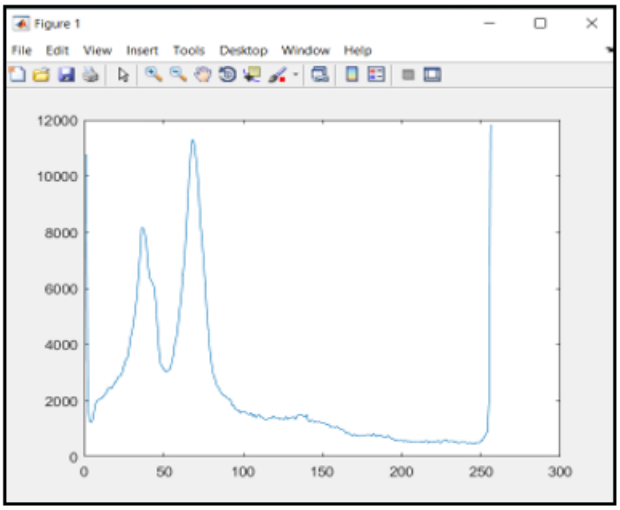
Histogramme

• L'histogramme d'image en niveau de gris

Une image en niveau de gris ne possède qu'une seule courbe, à gauche se situent les pixels noirs, à droite les pixels blancs, et au milieu, toutes les nuances de gris.



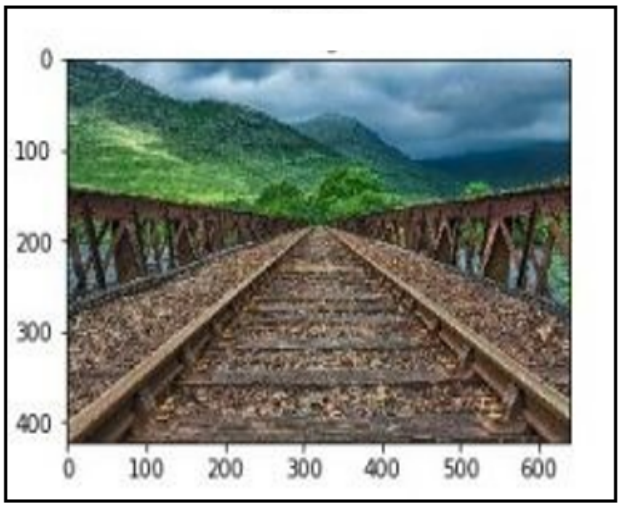
Image



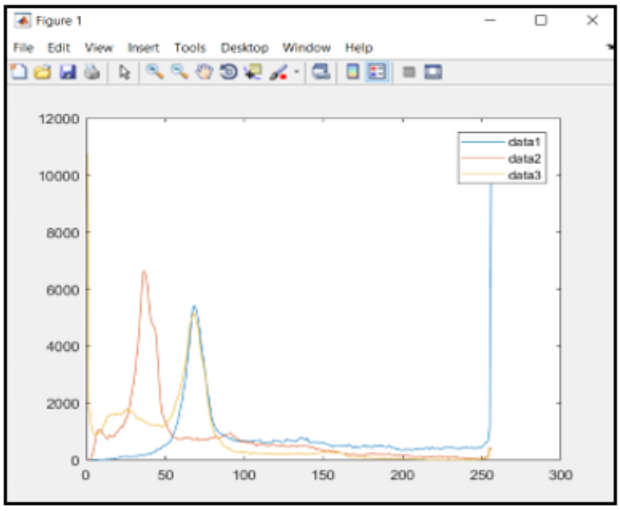
Histogramme

• L'histogramme d'image couleur

Si on a une image en couleur il nous faut maintenant avoir les nuances sur les trois canaux (Rouge, Vert et Bleu). On a donc 3 courbes.

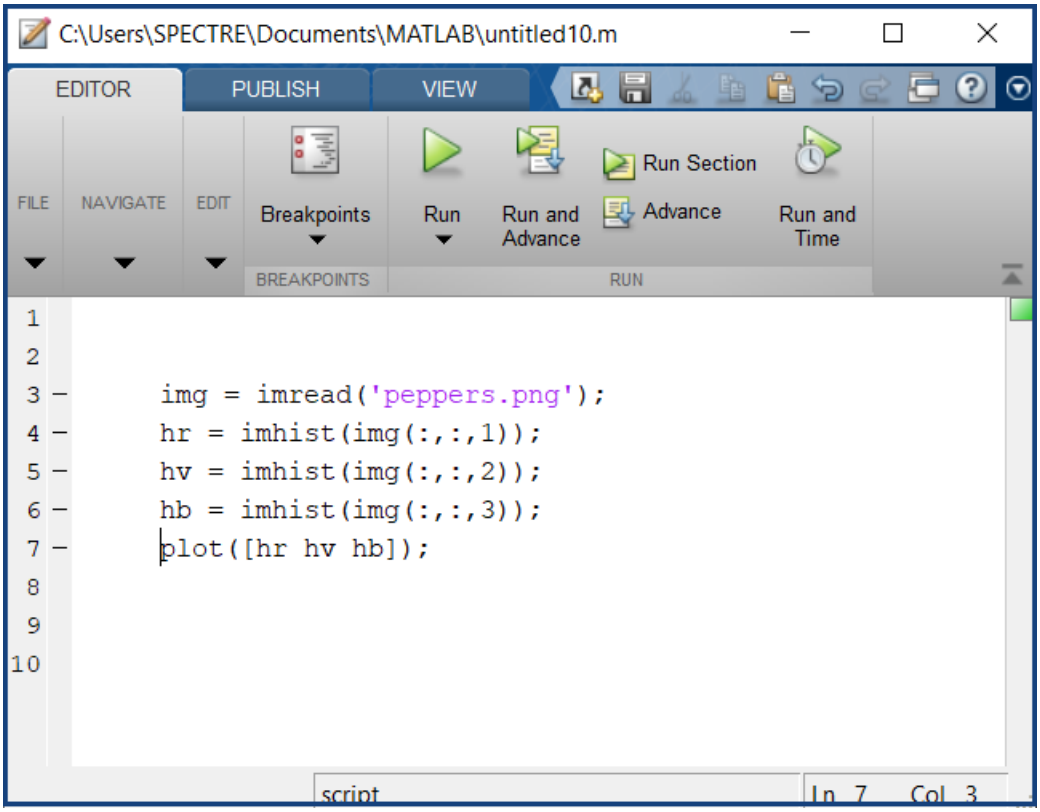


Image



Histogramme

⊸ **Code MATLAB :** (pour les images couleur)



Chapitre 2

Transformée de Fourier

La transformation de Fourier, inventée par le physicien et mathématicien français Joseph Fourier au 18e siècle, est un outil mathématique de traitement du signal et d'image qui permet de passer d'une représentation temporelle à une représentation fréquentielle d'image. Cette théorie est basée sur le fait que toute fonction périodique est décomposable sur une base de sinus et de cosinus. Ainsi, on peut passer d'une représentation temporelle (dans le repère temporel classique) à une représentation en fréquence sur une base de sinus et de cosinus (dans le repère fréquentiel).

2.1 Transformée de Fourier continue : TFC

2.1.1 Définition et terminologie

Définition 2.1.1.

On considère une image monochrome (niveaux de gris) représentée par une fonction de deux variables réelles, à valeurs complexes, notée $f \in L^1(\mathbb{R}^2)$.
La transformée de Fourier de cette image est la fonction à deux variables réelles et à valeurs complexes définie par :

$$\mathcal{F}(f)(a,b) = \hat{f}(a,b) = \int_{\mathbb{R}} \int_{\mathbb{R}} f(x,y) e^{-i(ax+by)} dxdy. \tag{2.1}$$

La courbe d'équation $y = \hat{f}$ est appelée le spectre de f.

Remarque 2.1.1.

Pour f un signal (dans $\mathbb{R}$), on a $f \in L^1(\mathbb{R})$, sa transformée de Fourier est définie, pour $a \in \mathbb{R}$, par

$$\mathcal{F}(f)(a) = \hat{f}(a) = \int_{\mathbb{R}} f(x) e^{-iax} dx. \tag{2.2}$$

Remarque 2.1.2.

Nous pouvons rencontrer d'autres définitions équivalentes (notamment avec un 2π) de la transformée de Fourier.

Autrement dit : pour $f \in L^1(\mathbb{R}^2)$ et $(a,b) \in \mathbb{R}^2$

$$\mathcal{F}(f)(a,b) = \hat{f}(a,b) = \int_{\mathbb{R}}\int_{\mathbb{R}} f(x,y)\, e^{-2\pi i(ax+by)} dxdy, \tag{2.3}$$

ou

$$\mathcal{F}(f)(a,b) = \hat{f}(a,b) = (2\pi)^{-\frac{N}{2}} \int_{\mathbb{R}}\int_{\mathbb{R}} f(x,y) e^{-i(ax+by)} dxdy, \tag{2.4}$$

avec $N = 2$ dans le cas d'espace $\mathbb{R}^2$ et $N = 1$ dans $\mathbb{R}$.

N.B : Le passage de l'une a l'autre est évident, c'est un simple changement d'échelles

$$\hat{f}(s) = \hat{f}(2\pi s).$$

Exemple 2.1.1.

Considérons la ***fonction porte*** $\Pi_a(t) = \mathbb{1}_{[-\frac{a}{2},\frac{a}{2}]}(t)$ avec $a > 0$,

on a :

$$\widehat{\Pi}_a(\lambda) = \int_{\mathbb{R}} \Pi_a(t)\, e^{-2i\pi\lambda t} dt = \int_{-\frac{a}{2}}^{\frac{a}{2}} e^{-2i\pi\lambda t} dt.$$

Pour $\lambda \neq 0$

$$\widehat{\Pi}_a(\lambda) = [-\frac{1}{2i\pi\lambda}\, e^{-2i\pi\lambda t}]_{-\frac{a}{2}}^{\frac{a}{2}} = \frac{1}{\lambda\pi}\frac{e^{i\pi\lambda a} - e^{-i\pi\lambda a}}{2i} = \frac{\sin\lambda a\pi}{\lambda\pi}.$$

Pour $\lambda = 0$

$$\widehat{\Pi}_a(0) = \int_{-\frac{a}{2}}^{\frac{a}{2}} dt = a.$$

Exemple 2.1.2

On considère une fonction $f = \frac{1}{(2c)^2}\mathbb{1}_{[-c,c]^2}$, pour $c > 0$.

On a, $\forall (a,b) \in \mathbb{R}^2$,

$$\begin{aligned}\hat{f}(a,b) &= \int_{\mathbb{R}}\int_{\mathbb{R}} \frac{1}{(2c)^2}\mathbb{1}_{[-c,c]^2}(x,y) e^{-i(ax+by)} dxdy, \\ &= \Big(\int_{\mathbb{R}} \frac{1}{(2c)}\mathbb{1}_{[-c,c]}(x) e^{-iax} dx\Big)\Big(\int_{\mathbb{R}} \frac{1}{(2c)}\mathbb{1}_{[-c,c]}(y) e^{-iby} dy\Big).\end{aligned}$$

Et

$$\begin{aligned}\int_{\mathbb{R}} \frac{1}{(2c)}\mathbb{1}_{[-c,c]}(x) e^{-iax} dx &= \frac{1}{(2c)}\int_{-c}^{c} e^{-iax} dx, \\ &= \frac{1}{-2iac}(e^{-iac} - e^{iac}) = \frac{1}{(2c)}\sin(ac), \\ &= sinc(ac),\end{aligned}$$

avec

$$sinc(t) = \begin{cases} \frac{\sin(t)}{t} & t \neq 0 \\ 1 & t = 0. \end{cases}$$

On a donc

$$\hat{f}(a,b) = sinc(ac).sinc(bc).$$

Définition 2.1.2.

Si l'on écrit $f(\lambda)$ sous la forme polaire $\hat{f}(\lambda) = A(\lambda)e^{i\phi(\lambda)}$, avec $A(\lambda) > 0$ et $\phi(\lambda)$ défini modulo 2π, on utilise la terminologie suivante :

$$\begin{cases} A(\lambda)\text{est appelé } \textbf{le spectre} \text{ de} f. \\ A^2(\lambda)\text{est appelée } \textbf{l'énergie spectrale} \text{ de} f. \\ \phi(\lambda)\text{est appelée } \textbf{la phase spectrale} \text{ de} f. \end{cases}$$

2.1.2 Propriétés communes à toute transformée de Fourier

La transformée de Fourier de n'importe quelle fonction intégrable a des propriétés caractéristiques que nous énonçons dans la proposition suivante.

Proposition 2.1.1.

Pour toute fonction $f \in L^1(\mathbb{R})$ (resp. $f \in L^1(\mathbb{R}^2)$) sa transformée de Fourier vérifie :

i) $\int_{\mathbb{R}} f(x)dx$ existe.

ii) $\hat{f}(\lambda)$ est une fonction continue sur $\mathbb{R}$ (resp. sur $\mathbb{R}^2$).

iii) $\lim_{\lambda \to \pm\infty} \hat{f}(\lambda) = 0$ (i.e. f tend vers 0 à l'infini).

vi) Si on pose $\| \hat{f} \|_\infty = \sup_{\lambda \in \mathbb{R}} \hat{f}(\lambda)$ on a $\sup_{\lambda \in \mathbb{R}} | \hat{f}(\lambda) | \leq \| f \|_1$.

Pour démontrer (vi) on majore

$$| \hat{f}(\lambda) | \leq \int_{\mathbb{R}} | f(t)e^{-2i\pi\lambda t} | dt = \int_{\mathbb{R}} | f(t) | dt = \| f \|_1.$$

Cette majoration étant vraie quelque soit $\lambda \in \mathbb{R}$, on déduit bien que $\sup_{\lambda \in \mathbb{R}} | \hat{f}(\lambda) | \leq \| f \|_1$.■

2.1.3 Propriétés élémentaires de transformée de Fourier

Présentons les propriétés élémentaires fort utiles de la transformée de Fourier $\mathcal{F}(.)$.

1 - La linéarité :

Soient f , g deux fonctions satisfaisant les conditions d'existence de leurs transformées de Fourier respectives. Alors, pour tout $(a, b) \in \mathbb{R}^2$,

$$\mathcal{F}(af + bg) = a\mathcal{F}(f) + b\mathcal{F}(g) = a\hat{f} + b\hat{g}.$$

2 - Changement d'échelle :

Soit $f \in L^1(\mathbb{R})$, pour tout $a \in \mathbb{R}^*$, on a :

$$\mathcal{F}(f(ax))(s) = \frac{1}{|a|}\hat{f}(\frac{s}{a}).$$

3 - Translation temporelle :

Soit $f \in L^1(\mathbb{R})$, pour tout $a \in \mathbb{R}$, on a :

$$\mathcal{F}(f(x-a))(s) = e^{-ias}\hat{f}(s).$$

4 - Modulation (= translation fréquentielle) :

Soit $f \in L^1(\mathbb{R})$, pour tout $a \in \mathbb{R}$, on a :

$$\mathcal{F}(e^{iax}f(x))(s) = \hat{f}(s-a), \quad s \in \mathbb{R}.$$

La notation $\mathcal{F}(e^{iax}f(x))$ représente la transformée de Fourier de la fonction $x \longmapsto e^{iax}f(x)$.

5 - Transformée d'une dérivée :

Si f est intégrable sur $\mathbb{R}$ et dérivable et si sa dérivée $\frac{df}{dt}$ est intégrable sur $\mathbb{R}$ alors la transformée de Fourier de la dérivée de f est :

$$\mathcal{F}(\frac{df}{dx})(s) = is\mathcal{F}(f)(s), \quad s \in \mathbb{R}.$$

Par récurrence pour f de classe C^k et si chaque fonction dérivée est intégrable sur $\mathbb{R}$, on obtient :

$$\mathcal{F}(f^{(k)})(s) = (is)^k\mathcal{F}(f)(s), \quad s \in \mathbb{R}.$$

6 - Règle de multiplication par t :

Si la fonction $xf(x)$ appartient à $L^1(\mathbb{R})$, alors on a :

$$\frac{d}{ds}(\mathcal{F}(f))(s) = -i\mathcal{F}(xf(x))(s).$$

La notation $\mathcal{F}(xf(x))$ représente la transformée de Fourier de la fonction $x \longmapsto xf(x)$.

7 - Parité :

Soit $f(x)$ une fonction à valeurs dans $\mathbb{R}$. On montre immédiatement que :

► Si $f(x)$ est réelle paire alors $\hat{f}(s)$ est réelle paire.
En effet, on a dans ce cas :

$$\hat{f}(s) = \int_{-\infty}^{+\infty} f(x)e^{-isx}dx = 2\int_{0}^{+\infty} f(x)\cos(sx)\,\mathrm{d}x.$$

► Si $f(x)$ est réelle impaire alors $\hat{f}(s)$ est imaginaire (pure) impaire.
En effet, on a dans ce cas :

$$\hat{f}(s) = \int_{-\infty}^{+\infty} f(x)e^{-isx}dx = -2i\int_{0}^{+\infty} f(x)\sin(sx)\,\mathrm{d}x.$$

2.1.4 Tables des transformées de Fourier

Soit f une fonction de $L^1(\mathbb{R})$. On a le tableau suivant :

Fonction f	**Transformée de Fourier**
$f = \mathbb{1}_{[-a,a]}\ ,\ a \in \mathbb{R}_+^*$	$\dfrac{\sin(2\pi ap)}{\pi p}$
$f = \mathbb{1}_{[a,b]}\ ,\ a < b$	$\dfrac{\sin(\pi(b-a))}{\pi p}e^{-2i\pi p\frac{a+b}{2}}$
$f(x) = \dfrac{x^k}{k!}e^{-ax}\mathbb{1}_{[0,+\infty[}(x), a > 0$	$\dfrac{1}{(a+2i\pi p)^{k+1}}$
$f(x) = \dfrac{x^k}{k!}e^{ax}\mathbb{1}_{]-\infty,0]}(x), a > 0$	$-\dfrac{1}{(2i\pi p-a)^{k+1}}$
$f(x) = e^{-a\lvert x\rvert}, a > 0$	$\dfrac{2a}{a^2+4\pi^2p^2}$
$f(x) = (1-\lvert x\rvert)\mathbb{1}_{[-1,1]}(x)$	$\dfrac{\sin^2(\pi p)}{\pi^2p^2}$
$f(x) = \dfrac{1}{\sigma\sqrt{\pi}}e^{-\frac{x^2}{\sigma^2}}$	$e^{-\sigma^2\pi^2p^2}$

2.2 Le produit de convolution

On introduit ici quelques concepts mathématiques fondamentaux dont on a besoin par la suite.

2.2.1 Définitions

Définition 2.2.1.

Le produit de convolution noté (*) est un opérateur bilinéaire, un produit commutatif.

Définition 2.2.2.

Soient (f, g) deux fonctions définies dans $\mathbb{R}^2$ (ou $\mathbb{C}$). Si f et g sont intégrales sur $\mathbb{R}^2$, on définit :

$$(f * g)(x, y) = \int_{\mathbb{R}} \int_{\mathbb{R}} f(x - x', y - y')g(x', y')dxdy.$$

2.2.2 Propriétés de base

Soient f et g sont deux fonctions de $L^1(\mathbb{R})$.

1 - Commutativité :

Le produit de convolution est commutatif :

$$(f * g)(x, y) = (g * f)(x, y).$$

2 - Le Dirac δ est l'élément neutre de * :

Pour f dans $C_c^0(\mathbb{R})$, on a : $\forall x \in \mathbb{R}$,

$$(\delta * f)(x, y) = f(x, y).$$

3 - Translation :

Le produit de convolution est invariant par translation au sens suivant. Notons $\tau_h(.)$ la translation d'une fonction $f(x)$:

$$\tau_{(h,l)}(f(x, y)) = f(x - h, y - l),$$

on a alors

$$\tau_{(h,l)}(f) * g = \tau_{(h,l)}(f * g).$$

4 - Dérivée :

Si f et g sont $C^1(\mathbb{R})$ et si $f^{'}$ et $g^{'}$ sont dans $L^1(\mathbb{R})$, alors on a :

$$(f * g)' = f' * g = f * g'.$$

5 - Suite finie :

Soient $(u_{m,n})_{1\leq m\leq N, 1\leq n\leq N}$ et $(v_{m,n})_{1\leq m\leq N, 1\leq n\leq N}$, deux suites finies. On note $u * v$, le produit de convolution de u par v, la suite

$$(u * v)_{m,n} = \sum_{m'=1}^{N} \sum_{n'=1}^{N} u_{m-m',n-n'} v_{m',n'}.$$

2.3 Transformée de Fourier et convolution

Une propriété forte de $\mathcal{F}$ est de transformer le produit de convolution en une multiplication, et inversement. Cette propriété est centrale dans l'analyse des filtres notamment.

Proposition 2.3.1.

Pour $f, g \in L^1(\mathbb{R})$, on a :

$$\mathcal{F}((f * g)(x)) = \mathcal{F}(f(x)).\mathcal{F}(g(x)).$$

Autrement noté :

$$\widehat{(f * g)}(s) = \hat{f}(s).\hat{g}(s), \qquad \forall s \in \mathbb{R}.$$

Démonstration.

En vertu du théorème de Fubini (et en supposant que tous les termes intégrables sur $\mathbb{R}$).

On a

$$\widehat{(f * g)}(s) = \int_{\mathbb{R}} g(y) \, (\int_{\mathbb{R}} f(x - y) \exp(-ixs) dx) dy.$$

On effectue le changement de variable $z = (x - y)$,on obtient :

$$\widehat{(f * g)}(s) = \int_{\mathbb{R}} g(y) \, e^{-isy} \Big(\int_{\mathbb{R}} f(z) e^{-izs} dz \Big) dy.$$

Soit le résultat : $\widehat{(f * g)}(s) = \hat{f}(s).\hat{g}(s)$.

De manière similaire on montre que :

$$\widehat{f.g}(s) = \hat{f}(s) * \hat{g}(s)$$

Proposition 2.3.2

Soit $f, g \in L^1(\mathbb{R}^2)$, alors pour tout $a, b \in \mathbb{R}$, on a :

$$\widehat{(f * g)}(a, b) = \hat{f}(a, b) * \hat{g}(a, b)$$

Démonstration.

Pour tous $a, b \in \mathbb{R}$, on a :

$$\begin{aligned} \widehat{f * g}(a, b) &= \int_{\mathbb{R}} \int_{\mathbb{R}} f * g(x, y) e^{-i(ax+by)} dx dy, \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} \Big(\int_{\mathbb{R}} \int_{\mathbb{R}} f(x - x', y - y') g(x', y') dx' dy' \Big) e^{-i(ax'+by')} dx dy. \end{aligned}$$

Comme la fonction $|f(x - x', y - y'||g(x', y')|$ est intégrable sur $\mathbb{R}^4$, on peut modifier l'ordre dans lequel on effectue les intégrations et obtenir

$$\widehat{(f * g)}(a, b) = \int_{\mathbb{R}} \int_{\mathbb{R}} \Big(\int_{\mathbb{R}} \int_{\mathbb{R}} f(x - x', y - y') e^{-i(a(x-x')+b(y-y'))} dx dy \Big) g(x', y') e^{-i(ax'+by')} dx' dy'.$$

On obtient après un changement de variable :

$$\int_{\mathbb{R}}\int_{\mathbb{R}} f(x-x',y-y')e^{-i(a(x-x')+b(y-y'))}dxdy = \int_{\mathbb{R}}\int_{\mathbb{R}} f(x,y)e^{-i(ax+by)}dxdy = \hat{f}(a,b).$$

On a donc finalement,

$$\widehat{(f*g)}(a,b) = \int_{\mathbb{R}}\int_{\mathbb{R}} \hat{f}(a,b)g(x',y')e^{-i(ax'+by')}dx'dy' = \hat{f}(a,b)\hat{g}(a,b).$$

Exemple 2.3.1

Soit à résoudre l'équation différentielle linéaire du second ordre, avec un second membre f (supposée dans $L^1(\mathbb{R})$)

$$-\frac{1}{\omega^2}g'' + g = f.$$

Formellement, si on applique la transformée de Fourier aux deux membres de cette équation, on obtient :

$$-\frac{1}{\omega^2}(2i\pi t)^2\,\hat{g}(t) + \hat{g}(t) = \hat{f}(t),$$

soit encore

$$\Big(1 + \frac{4\pi^2t^2}{\omega^2}\Big)\hat{g}(t) = \hat{f}(t).$$

Comme la fonction $1 + \dfrac{4\pi^2t^2}{\omega^2} > 0$ dans $\mathbb{R}$, on déduit que

$$\hat{g}(t) = \frac{1}{(1+\frac{4\pi^2t^2}{\omega^2})}\hat{f}(t).$$

En utilisant la table des transformées de Fourier,

on vérifie que $\dfrac{1}{(1+\frac{4\pi^2t^2}{\omega^2})} = \hat{h}(t)$ avec $h(x) = \dfrac{\omega}{2}e^{-\omega|x|}$.

Finalement, on peut écrire

$$\hat{g}(t) = \hat{h}(t)\hat{f}(t).$$

Comme ici, f et h sont dans $L^1(\mathbb{R})$, alors on a $\hat{h}(t)\hat{f}(t) = \widehat{(h*f)}(t)$ et grâce à l'injectivité de la transformée de Fourier $g = h*f$ p.p ou encore

$$g(x) = \frac{\omega}{2}\int_{\mathbb{R}} e^{-\omega|x-t|}f(t)dt \text{ , p.p} \quad \forall x \in \mathbb{R}.$$

2.4 Transformée de Fourier inverse

Sous certaines conditions, il est possible d'inverser la transformée de Fourier, c'est-à-dire de retrouver f en connaissant $\hat{f}$.

Définition 2.4.1

Soit f une fonction de $L^1(\mathbb{R}^2)$, Si sa transformée de Fourier $\hat{f}$ est aussi dans $L^1(\mathbb{R}^2)$, on a alors pour presque tout $(x, y) \in \mathbb{R}^2$,

$$\mathcal{F}^{-1}(\hat{f}(a,b)) = f(x,y) = \int_{\mathbb{R}}\int_{\mathbb{R}} \hat{f}(a,b)\, e^{i(ax+by)} da\, db. \tag{2.5}$$

Elle possède des propriétés tout à fait analogues à celles de la transformée de Fourier.

Remarque 2.4.1

Dans $\mathbb{R}$, on a : $\quad f(x) = \int_{\mathbb{R}} \hat{f}(a) e^{iax} da.$

Exemple 2.4.1

On considère une fonction $f(t) = e^{-|t|}$, on a facilement par un calcul direct $\hat{f}(\lambda) = \dfrac{2}{1+\lambda^2}$.

La fonction $\hat{f} \in L^1(\mathbb{R})$. De plus, f est une fonction continue sur $\mathbb{R}$, on a :

$$f(t) = \int_{\mathbb{R}} \hat{f}(\lambda) e^{i\lambda t} \mathrm{d}\lambda = \int_{-\infty}^{+\infty} \frac{2}{1+\lambda^2} e^{i\lambda t}\, \mathrm{d}\lambda\ , \qquad \forall t \in \mathbb{R}.$$

Comme $\widehat{f}(\lambda)$ est une fonction paire, la partie d'intégrale en $\sin(\lambda)$ est nulle et on a :

$$\int_{-\infty}^{+\infty} \frac{2}{1+\lambda^2} e^{i\lambda t}\, \mathrm{d}\lambda = \int_{-\infty}^{+\infty} \frac{2}{1+\lambda^2} \cos(\lambda)\, \mathrm{d}\lambda = \int_{0}^{+\infty} \frac{4}{1+\lambda^2} \cos(\lambda)\, \mathrm{d}\lambda = e^{-|t|}, \qquad \forall t \in \mathbb{R}.$$

Proposition 2.4.1. (Injectivité de la transformée de Fourier)

Soient f, g dans $L^1(\mathbb{R})$, alors $\quad \widehat{f}(\lambda) = \widehat{g}(\lambda),\ \ \forall \lambda \in \mathbb{R} \ \Rightarrow\ f = g$ p.p.

Deux fonctions intégrables, ayant même transformée de Fourier sont égales presque partout.

Démonstration.

C'est une conséquence simple du théorème d'inversion précédent.
Posons $h = f - g$, on a $h \in L^1(\mathbb{R})$ et par hypothèse $\hat{h} = 0$. Comme $\hat{h} \in L^1(\mathbb{R})$, on peut appliquer la définition et

$$h(x) = \int_{\mathbb{R}} \widehat{h}(\lambda) e^{i\lambda x} d\lambda = 0.$$

Ce qui s'écrit encore $\quad f = g \quad$ p.p. ■

N.B : De ce fait, la transformée de Fourier ne s'applique plus telle quelle. Il faut donc la passer dans le domaine discret.

2.5 Transformée de Fourier Discrète : TFD

La transformation de Fourier continue n'est pas très utile pour analyser des signaux complexes quand l'intégrale n'a pas de solution analytique fermée. La transformation de Fourier discrète est une alternative qui permet d'analyser les images numériques (discrétisés).

La transformation de Fourier discrète (en anglais on parle de Discrete Fourier Transform (DFT)) est un outil mathématique de traitement d'image numérique, qui est l'équivalent discret de la transformation de Fourier continue qui est utilisée pour le traitement d'image analogique.

Dans cette partie, nous allons définir les notions analogues à celles que nous avons déjà vues pour des fonctions analogiques, mais pour des suites finies.

2.5.1 Transformation discrète en 1D (signal)

Pour un signal de N échantillons.

Définition 2.5.1.

La transformée de Fourier discrète transforme une séquence $(w_m)_{1\leq m\leq N}$ de N nombres complexes en une autre séquence de nombres complexes $(\hat{w}_k)_{1\leq k\leq N}$, qui définie par :

$$\hat{w}_k = \sum_{m=1}^{N} w_m e^{-2i\pi\frac{km}{N}} \qquad pour \qquad 0 \leq k < N, \tag{2.6}$$

$$= \sum_{m=1}^{N} w_m \big[\cos(2\pi\frac{km}{N}) - i\cdot\sin(2\pi\frac{km}{N})\big]. \tag{2.7}$$

Où la dernière expression découle de la première par **la formule d'Euler** .

2.5.2 Transformation inverse

La transformée de Fourier discrète est une transformation linéaire inversible. On peut reconstruire le signal $(w_m)_{1\leq m\leq N}$ à partir de sa représentation fréquentielle $(\hat{w}_k)_{1\leq k\leq N}$ par la formule :

$$w_m = \frac{1}{N}\sum_{k=1}^{N} \hat{w}_k e^{2i\pi k\frac{m}{N}}. \tag{2.8}$$

S'appelle la transformation de Fourier discrète inverse (IDFT).

En effet, on calcule :

$$A = \frac{1}{N}\sum_{k=1}^{N} \hat{w}_k e^{2i\pi k\frac{m}{N}} = \frac{1}{N}\sum_{k=1}^{N}\Big(\sum_{s=1}^{N} w_s e^{-2i\pi k\frac{s}{N}}\Big)\, e^{2i\pi k\frac{m}{N}},$$

$$A = \frac{1}{N}\sum_{s=1}^{N} w_s \Big(\sum_{k=1}^{N} e^{2i\pi k\frac{(m-s)}{N}}\Big),$$

$$\text{si} \quad i \neq n \quad \longmapsto \quad \sum_{k=1}^{N} e^{2i\pi k\frac{(m-s)}{N}} = \frac{1-e^{2i\pi(m-s)}}{1-e^{2i\pi k\frac{m-s}{N}}} = 0,$$
$$\text{si} \quad i = n \quad \longmapsto \quad \sum_{k=1}^{N} e^{2i\pi k\frac{(m-s)}{N}} = \sum_{k=1}^{N} 1 = N,$$

$$A = \frac{1}{N}\sum_{s=1}^{N} w_s \left(\sum_{k=1}^{N} e^{2i\pi k\frac{(m-s)}{N}}\right) = \frac{1}{N} w_m \; N = w_m \; \blacksquare$$

2.5.3 Les propriétés de la DFT

Les propriétés de la TFD sont similaires à celles enoncées pour la TFC.

1 - Linéarité :

Soit $(w_m)_{1\leq m\leq N}$ et $(v_n)_{1\leq n\leq N}$ deux suites finies de nombres réels, pour tout $(a,b) \in \mathbb{R}^2$,

$$\mathcal{F}(aw_m + bv_n) = a\hat{w}_k + b\hat{w}_l \qquad \text{pour} \quad 0 \leq k,l < N.$$

2 - Périodicité :

La périodicité peut être démontré directement à partir de la définition, pour $k = 1,\ldots,N$,

$$\begin{aligned}
\hat{w}_{k+N} &= \sum_{m=1}^{N} w_m e^{-2i\pi m\frac{k+N}{N}}, \\
&= \sum_{m=1}^{N} w_m e^{-2i\pi\frac{(km}{N}} + \sum_{m=1}^{N} w_m e^{-2i\pi m}, \\
&= \sum_{m=1}^{N} w_m e^{-2i\pi\frac{km}{N}}, \\
&= \hat{w}_k.
\end{aligned}$$

Où nous avons utilisé le fait que $e^{-2i\pi m} = 1 \quad \forall m$.

De la même manière il peut être démontré que la formule IDFT conduit à une extension périodique.

3 - Symétrie :

$$\begin{aligned}
\hat{w}_{-k} &= \sum_{m=1}^{N} w_m e^{-2i\pi m\frac{(-k)}{N}}, \\
&= \sum_{m=1}^{N} w_m \left(e^{-2i\pi\frac{km}{N}}\right)^*, \\
&= (\hat{w}_k)^*.
\end{aligned}$$

Où $\hat{w}_k^*$ désigne le complexe conjugué de $\hat{w}_k$.

Avec la propriété de périodicité, on peut dire que :

$$\hat{w}_{-k} = \hat{w}_{N-k} = \hat{w}_k^*.$$

4 - Interprétation matricielle :

Une autre façon de regarder la DFT est de noter que dans la discussion ci-dessus, la DFT peut être exprimée en version matricielle (**une matrice de Vandermonde**) :

$$\begin{pmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \vdots \\ \vdots \\ \hat{w}_N \end{pmatrix} = \begin{pmatrix} e^{\frac{-2.\pi.i}{N}} & e^{\frac{-4.\pi.i}{N}} & \cdots & \cdots & e^{-2.\pi.i} \\ e^{\frac{-4.\pi.i}{N}} & e^{\frac{-8.\pi.i}{N}} & \cdots & \cdots & e^{-4.\pi.i} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{-2.\pi.i} & e^{-4.\pi.i} & \cdots & \cdots & e^{-2.\pi.iN} \end{pmatrix} . \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_N \end{pmatrix}$$

5 - Convolution :

Soient h et w deux suites finies, définies sur $\{1, ..., N\}^2$. On a alors, pour tout $k \in \{1, ..., N\}^2$,

$$\widehat{(h * w)}_k = \widehat{(h)}_k \cdot \widehat{(w)}_k. \tag{2.9}$$

2.5.4 Transformation discrète en 2D (image)

Pour $(w_{m,n})$ une image de taille $N \times N$.

Définition 2.5.2.

Soit $(w_{m,n})_{1\leq m\leq N, 1\leq n\leq N}$ une suite finie de nombres réels. Sa transformée de Fourier discrète est définie pour $(k, l) \in \{1, ..., N\}^2$ (ou, de façon équivalente, sur $(k, l) \in \{\frac{-N}{2} + 1, ..., \frac{N}{2}\}^2$) par

$$\hat{w}_{k,l} = \sum_{m,n=1}^{N} w_{m,n} e^{-2i\pi \frac{km+ln}{N}} = \sum_{m=1}^{N} \sum_{n=1}^{N} w_{m,n} e^{-2i\pi \frac{km+ln}{N}}. \tag{2.10}$$

Remarque 2.5.1.

1. Il n'est pas difficile de voir que $\hat{w}_{k,l} = \hat{w}_{k+t_1 N, l+t_2 N}$, $\forall (t_1, t_2) \in \mathbb{Z}^2$.

Puisque $\forall (m, n) \in \mathbb{Z}^2$, $e^{-2i\pi \frac{km+ln}{N}} = e^{-2i\pi \frac{(k+t_1 N)m+(l+t_2 N)n}{N}}$.

2. Comme dans le cas de la transformée de Fourier, le fait que la suite w soit à valeur

dans $\mathbb{R}$ implique que : $\hat{w}^*_{k,l} = \hat{w}_{-k,-l}$ ■

L'inversion de la transformée de Fourier discrète est donnée par

Proposition 2.5.1. (Inversion de la transformée de Fourier)

Soit $(w_{m,n})_{1\leq m\leq N, 1\leq n\leq N}$ une suite finie de nombres réels. Si on note $(\hat{w}_{k,l})_{1\leq k\leq N, 1\leq l\leq N}$, ses coefficients de Fourier, on a alors, pour tout $(m, n) \in \{1, ..., N\}^2$,

$$w_{m,n} = \frac{1}{N^2} \sum_{k,l=1}^{N} \hat{w}_{k,l} e^{2i\pi \frac{km+ln}{N}} = \frac{1}{N^2} \sum_{k=1}^{N} \sum_{l=1}^{N} \hat{w}_{k,l} e^{2i\pi \frac{km+ln}{N}}. \tag{2.11}$$

2.5.5 Algorithme de la transformée de Fourier lente

La méthode basique de calcul de la TFD d'une image (l'implémentation naïve) consiste à calculer les N^2 Valeurs $\hat{w}_{k,l}$:

$$\hat{w}_{k,l} = \sum_{m,n=1}^{N} w_{m,n} e^{-2i\pi \frac{km+ln}{N}}.$$

et ceci pour $(k,l) \in [1,N]$.

On peut construire un algorithme exploitant directement la formule. On obtient alors l'Algorithme 1. Au cours de l'exécution de cet algorithme,

- On initialise N^2 fois une variable à 0,
- On calcule N^4 fois l'arguments des cosinus et sinus,
- On calcule N^4 fois les cosinus et sinus,
- On calcule N^4 fois les produits et les sommes impliquant $\hat{w}_{k,l}^{\text{réelle}}$, $\hat{w}_{k,l}^{\text{imaginaire}}$, $w_{m,n}$, le cosinus et le sinus.

Algorithme 1 : Transformée de Fourier lente

Entrées :
w : image de taille $N \times N$;
Sorties :
$\hat{w}^{*\text{réelle}}$, $\hat{w}^{*\text{imaginaire}}$: images de taille $N \times N$ contenant les parties réelles et imaginaires de la transformée de Fourier de w ;
Début;
pour $k = 1, ..., N$ **faire**
  pour $l = 1, ..., N$ **faire**
    $\hat{w}_{k,l}^{\text{réelle}} = \hat{w}_{k,l}^{\text{imaginaire}} = 0$;
    pour $m = 1, ..., N$ **faire**
      pour $n = 1, ..., N$ **faire**
        $\hat{w}_{k,l}^{\text{réelle}} = \hat{w}_{k,l}^{\text{réelle}} + w_{m,n} \cos(-2\pi \frac{km+ln}{N})$;
        $\hat{w}_{k,l}^{\text{imaginaire}} = \hat{w}_{k,l}^{\text{imaginaire}} + w_{m,n} \sin(-2\pi \frac{km+ln}{N})$;
      fin
    fin
  fin
fin

La complexité temporelle de cet algorithme est donc $\mathcal{O}(N^4)$ pour une image ($\mathcal{O}(N^2)$ pour un signal). Le nombre d'échantillons est de l'ordre de plusieurs milliers à plusieurs millions et cette méthode conduit à des calculs extrêmement longs.

2.6 Transformée de Fourier rapide : FFT

Ce paragraphe se veut directement tourné vers les applications informatiques de la TFD.

On appelle transformée de Fourier rapide (acronyme anglais : FFT ou fast Fourier transform), un algorithme permettant de réduire le nombre d'opérations, en particulier le nombre de multiplications, pour calculer la transformée de Fourier discrète (DFT) d'une séquence ou son inverse (IDFT) dont la complexité est de l'ordre de $\mathcal{O}(N^2.log_2(N))$, pour une image de taille $N \times N$. Le coût de l'algorithme de la FFT est moins faible que celui de la TFD.

En tant qu'algorithme, il existe donc plusieurs méthodes pour calculer la FFT. Mais dans ce mémoire, nous allons étudier celui du Cooley-Tukey.

2.6.1 L'algorithme de Cooley-Tukey

L'algorithme de la transformée de Fourier rapide a été initialement découvert par Gauss en 1805, mais il n'a eu de succès qu'en 1965 après la publication [4] de celui-ci par J. W. Cooley et J. W. Tukey. C'est pour cette raison que l'algorithme de base porte habituellement leurs noms.

La transformée de Fourier rapide a été « décrétée » l'un des 10 algorithmes majeurs du 20^e siècle [2] établie par la version américaine de la société de mathématiques appliquées et industrielles.

L'algorithme de Cooley-Tukey, appelé aussi algorithme de réduction à base 2 dans le domaine temporel, s'applique seulement dans le cas où le nombre N d'échantillons $w_{m,n}$ s'exprime sous la forme 2^K et permet alors de simplifier le problème par une décomposition dichotomique.

L'idée derrière cet algorithme est la stratégie « **divide and conquer**»
(" diviser pour mieux régner "), c'est-à-dire on divise un problème en plusieurs sous-problèmes de façon récursive pour gagner en temps de calcul, il factorise la DFT pour réduire le nombre d'opérations de N^4 à $N^2.log_2(N)$.

Exemple 2.6.1. (Comparaison TFD et FFT)

Un calcul de transformée de Fourier discrète est un calcul nécessite N^4 multiplications/additions de nombres complexes.
Si on suppose qu'un calculateur effectue 10^9 opérations par seconde.

◇ Un calcul de transformée sur un signal de $N = 10^3$ échantillons nécessitera 10^{-3}s.

◇ Un calcul sur une image de taille $N \times N = 10^6$ nécessitera N^4 soit 10^{12} opérations et une quinzaine de minutes de calcul.

◇ Si on envisage de traiter des données dans un domaine à trois dimensions (sur des vecteurs de taille $N \times N \times N$) il faudrait alors N^6 soit 10^{18} opérations, ce qui nécessite quelques dizaines d'années.

La transformée de Fourier rapide réduit considérablement le nombre d'opérations à effectuer : au lieu d'effectuer N^4 opérations, il suffira d'en faire $N^2 \log_2 N$ (pour une image).

Dans les trois exemples précédents on aura à faire 10^4 **;** 2×10^7 **et** 3×10^{10} **opérations** ce qui nécessitera respectivement $10^{-5}s$**,** $2 \times 10^{-2}s$ **et** 30**s** ... □

2.6.2 Transformée de Fourier rapide en dimension 1 (un signal)

Pour simplifier la présentation, nous nous contentons de décrire l'algorithme de Cooley-Tukey dans le cas d'un signal (1D) avec $w \in \mathbb{R}^N$ dont la taille N est une puissance de 2.

Pour expliquer cet algorithme, nous utiliserons la récursivité en montrant que le calcul d'une transformée de Fourier de taille N se ramène au calcul de deux transformées de Fourier de taille $N/2$ suivi de $N/2$ multiplications.

On part donc de la formulation de la TFD de base vu dans les paragraphes précédents :

$$\hat{w}_k = \sum_{m=1}^{N} w_m e^{-2i\pi \frac{km}{N}}.$$

On peut décomposer cette somme en deux sous-sommes.
La première somme les termes pairs et la seconde les termes impairs, on obtient (en supposant que N soit pair "c'est une puissance de 2"), pour $m \leq N/2$:

$$\hat{w}_k = \sum_{m=1}^{\frac{N}{2}} w_{2m} e^{-2i\pi \frac{k \times 2m}{N}} + \sum_{m=1}^{\frac{N}{2}} w_{2m-1} e^{-2i\pi \frac{k \times (2m-1)}{N}}.$$

En développant un peu le calcul dans les exponentielles :

$$\hat{w}_k = \sum_{m=1}^{\frac{N}{2}} w_{2m} e^{-2i\pi \frac{km}{N/2}} + \sum_{m=1}^{\frac{N}{2}} w_{2m-1} e^{\frac{2i\pi k}{N}} e^{-2i\pi \frac{km}{N/2}}.$$

On peut donc sortir de la somme l'exponentielle ne dépendant que de "k" :

$$\hat{w}_k = \sum_{m=1}^{\frac{N}{2}} w_{pair} e^{-2i\pi \frac{km}{N/2}} + e^{\frac{2i\pi k}{N}} \sum_{m=1}^{\frac{N}{2}} w_{impair} e^{-2i\pi \frac{km}{N/2}}.$$

Et là, on remarque que les deux sommes ne sont, en réalité, que les transformées de Fourier de la séquence des termes pairs de w (noté $\hat{w}^{\text{impair}}$) et de la séquence des termes impairs de w (noté $\hat{w}^{\text{pair}}$).

On a donc :

$$\hat{w}_k = \hat{w}_k^{\text{pair}} + e^{\frac{2i\pi k}{N}} \hat{w}_k^{\text{impair}} \quad , \quad 0 \leq k < \tfrac{N}{2}.$$

Ici, on détecte un petit problème : **comment faire si** m **est supérieur à** $N/2$ **?** Et bien, on applique tout simplement la propriété de périodicité de la transformée de Fourier discrète.

De même, si on calcule $\hat{w}_{k+N/2}$ on a :

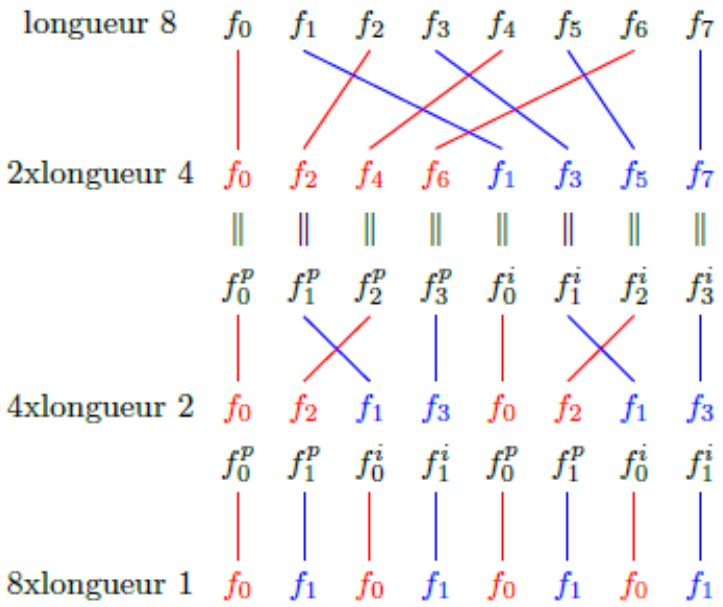
$$\begin{aligned}
\hat{w}_{k+\frac{N}{2}} &= \sum_{m=1}^{N} w_m e^{-2i\pi m\frac{k+N/2}{N}}, \\
&= \sum_{m=1}^{\frac{N}{2}} w_{2m} e^{-2i\pi\frac{(k+N/2)\times 2m}{N}} + \sum_{m=1}^{\frac{N}{2}} w_{2m-1} e^{-2i\pi\frac{(k+N/2)\times(2m-1)}{N}}, \\
&= \sum_{m=1}^{\frac{N}{2}} w_m^{\text{pair}} e^{-2i\pi m} e^{-2i\pi\frac{km}{N/2}} + e^{\frac{2i\pi k}{N}} \sum_{m=1}^{\frac{N}{2}} w_m^{\text{impair}} e^{-2i\pi m} e^{-2i\pi\frac{km}{N/2}}, \\
&= \sum_{m=1}^{\frac{N}{2}} w_m^{\text{pair}} e^{-2i\pi\frac{km}{N/2}} - e^{\frac{2i\pi k}{N}} \sum_{m=1}^{\frac{N}{2}} w_m^{\text{impair}} e^{-2i\pi\frac{km}{N/2}}, \\
&= \hat{w}_k^{\text{pair}} - e^{2i\pi k/N}\hat{w}_k^{\text{impair}}.
\end{aligned}$$

Finalement, on obtient pour tout $k \in \{1, ..., \frac{N}{2}\}$, l'expression récursive de la transformée de Fourier discrète :

$$\begin{cases}
\hat{w}_k &= \hat{w}_k^{\text{pair}} + e^{2i\pi\frac{k}{N}}\hat{w}_k^{\text{impair}}, \\
\hat{w}_{k+\frac{N}{2}} &= \hat{w}_k^{\text{pair}} - e^{2i\pi\frac{k}{N}}\hat{w}_k^{\text{impair}}.
\end{cases}$$

Cela signifie qu'en calculant deux transformées de Fourier de longueur $N/2$, on est capable de calculer deux éléments d'une transformée de Fourier de longueur N. En supposant pour simplifier que N soit une puissance de deux, cela dessine naturellement une implémentation récursive de la FFT.

Exemple 2.6.2. Décomposition par FFT d'un vecteur de longueur 8 ($w = f$).



On peut donc construire l'algorithme récursif de la FFT.

Algorithme 2 : Transformée de Fourier rapide, pour un signal

Entrées :
w : signal de taille N (puissance de 2) ;
Sorties :
$\hat{w}$: signal de taille N contenant la transformée de Fourier de w ;
Début;
si $N = 1$ **alors**
  $\hat{w}_1 = w_1$;
sinon
  extraire w^{impair} et w^{pair} ;
  calculer $\hat{w}^{\text{impair}}$ et $\hat{w}^{\text{pair}}$ avec l'algorithme de FFT;
  pour $k = 1, ..., \dfrac{N}{2}$ **faire**
    $\hat{w}_k = \hat{w}_k^{\text{pair}} + e^{\frac{2i\pi k}{N}} \hat{w}_k^{\text{impair}}$;
    $\hat{w}_{\frac{N}{2}+k} = \hat{w}_k^{\text{pair}} - e^{\frac{2i\pi k}{N}} \hat{w}_k^{\text{impair}}$;
  fin
fin

Donc l'algorithme 2 sera de complexité $\mathcal{O}(N \log_2 N) \ll \mathcal{O}(N^2)$, ce qui est bien mieux que le premier algorithme naïf que nous avons implémenté.

2.6.3 Transformée de Fourier rapide en dimension 2 (une image)

Soit $(w_{m,n})_{1\leq m\leq N, 1\leq n\leq N}$ une image. Pour calculer sa transformée de Fourier, on décompose simplement pour $(k,l) \in \{1, ..., N\}^2$,

$$\begin{aligned}
\hat{w}_{k,l} &= \sum_{m,n=1}^{N} w_{m,n} e^{-2i\pi \frac{km+ln}{N}}, \\
&= \underbrace{\sum_{m=1}^{N} \left(\underbrace{\sum_{n=1}^{N} w_{m,n}\, e^{-2i\pi \frac{ln}{N}}}_{\text{DFT du signal 1D } (w_{m,n})_{1\leq n\leq N}} \right) e^{-2i\pi \frac{km}{N}}}_{\text{DFT du signal 1D } (\sum_{n=1}^{N} w_{m,n} e^{-2i\pi \frac{ln}{N}})_{1\leq m\leq N}}.
\end{aligned}$$

Pour chaque $m \in \{1, ..., N\}$, $\quad l \longmapsto \displaystyle\sum_{n=1}^{N} w_{m,n} e^{-2i\pi \frac{ln}{N}}$ est simplement la transformée de Fourier 1D du signal $(w_{m,n})_{1\leq n\leq N} \in \mathbb{R}^N$.

De même, pour l fixé, la somme en m correspond à la transformée de Fourier 1D de

$$m \longmapsto \sum_{n=1}^{N} w_{m,n} e^{-2i\pi \frac{ln}{N}}.$$

Finalement, on peut calculer la transformée de Fourier 2D d'une image à l'aide de l'algorithme décrit dans l'Algorithme 3.

Algorithme 3 : Transformée de Fourier rapide, pour une image

Entrées :
w : image de taille $N \times N$ (N puissance de 2) ;
Sorties :
$\hat{w}$: image de taille $N \times N$ contenant la transformée de Fourier de w ;
Début;
pour $m = 1, ..., N$ **faire**
  extraire la ligne (un signal 1D) $(w_{m,n})_{1 \leq n \leq N}$;
  calculer la FFT 1D de ce signal;
  ranger cette transformée de Fourier à la ligne m d'une image $(v_{m,l})_{1 \leq m,l \leq N}$;
fin
pour $l = 1, ..., N$ **faire**
  extraire la colonne (un signal 1D)$(v_{m,l})_{1 \leq m,l \leq N}$;
  calculer la FFT 1D de ce signal;
  ranger cette transformée de Fourier à la ligne l de $\hat{w}$;
fin

Remarque 2.6.1.

On a bien sûr une construction et un algorithme analogue pour la transformée de Fourier 2D inverse.

Chapitre 3

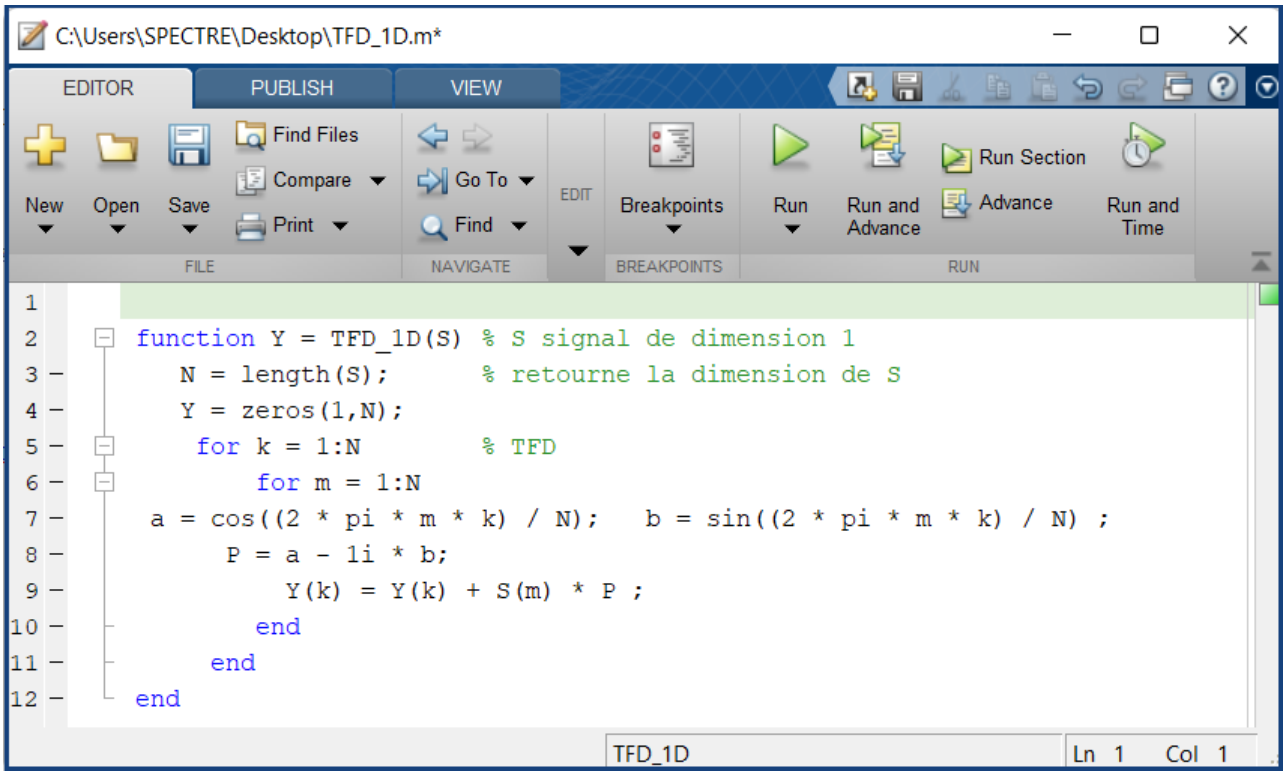
Évaluation de la transformée Fourier avec MATLAB

Dans ce chapitre, nous allons implémenter l'algorithme de TFD en partant d'une implémentation naïf, puis une implémentation rapide pour essayer de calculer la transformée de Fourier d'une image de la manière la plus efficace possible. Nous utiliserons pour cela le programme MATLAB.

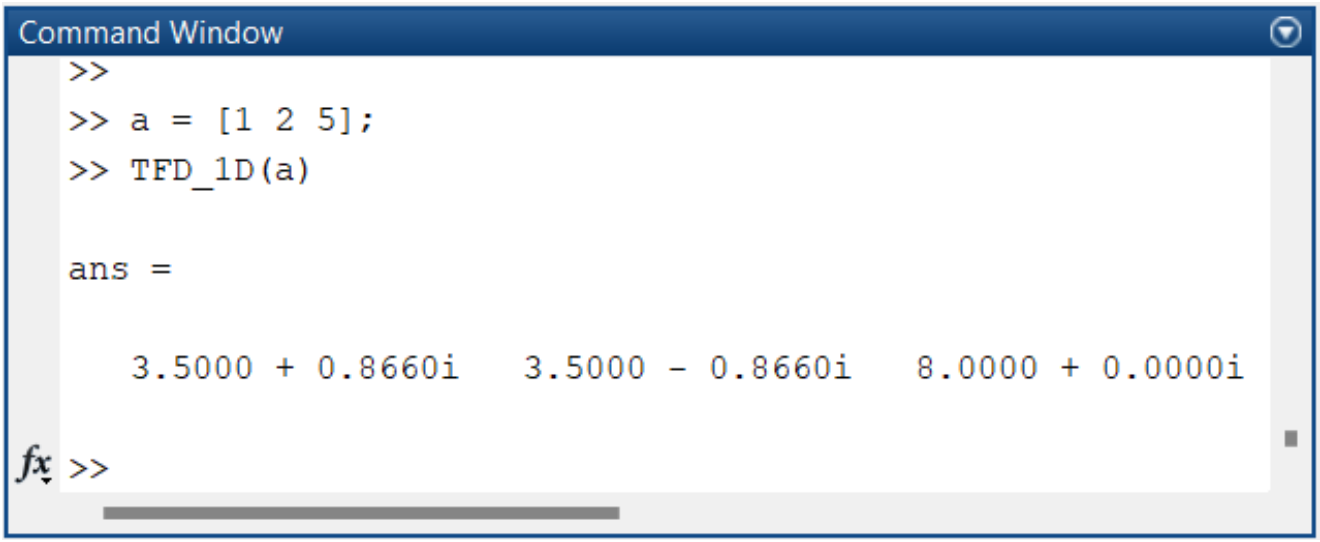
MATLAB (« matrix laboratory ») est un langage de script émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs et peut s'interfacer avec d'autres langages comme le C, C++,...

3.1 Transformée de Fourier lente (1D) :

Code d'exécution :



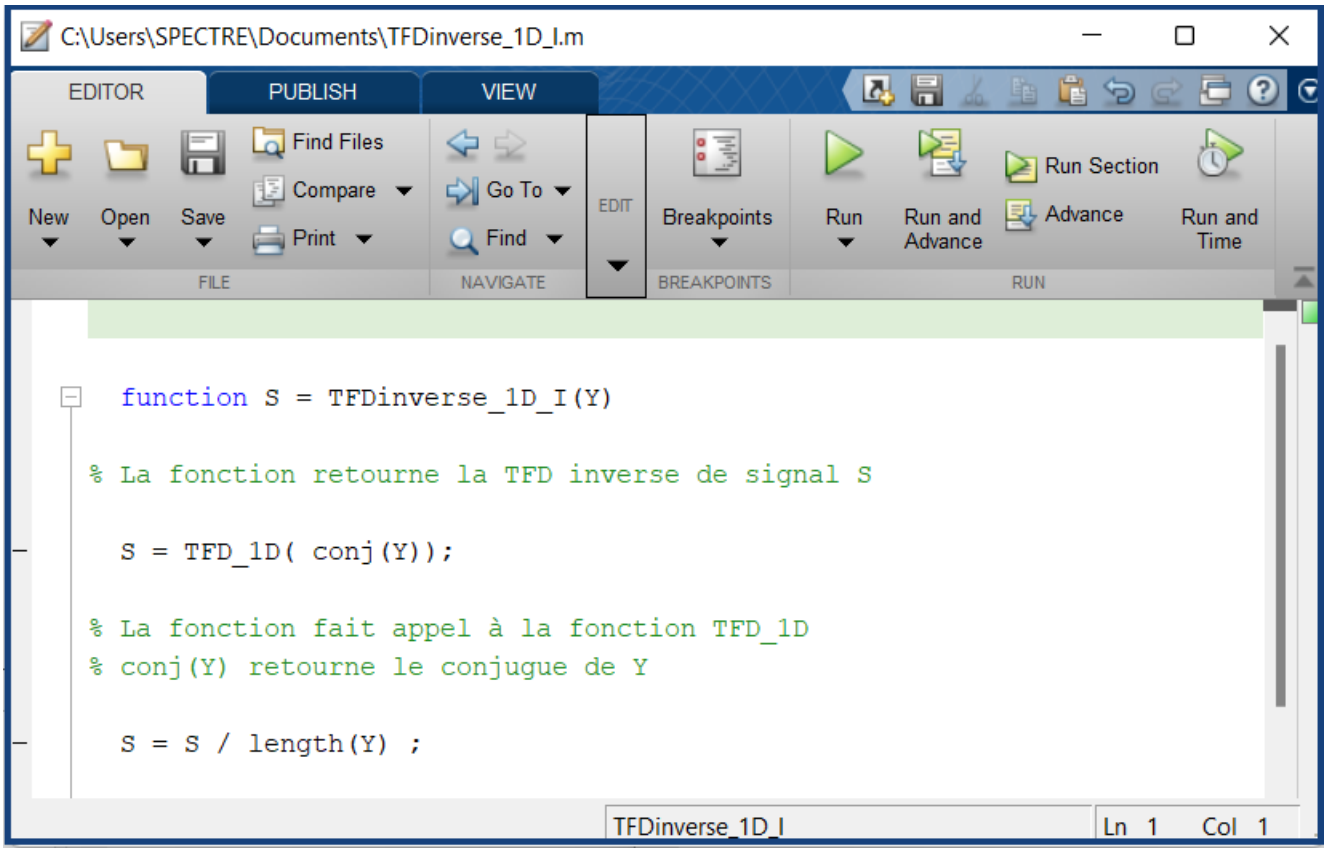
Exemple d'application :



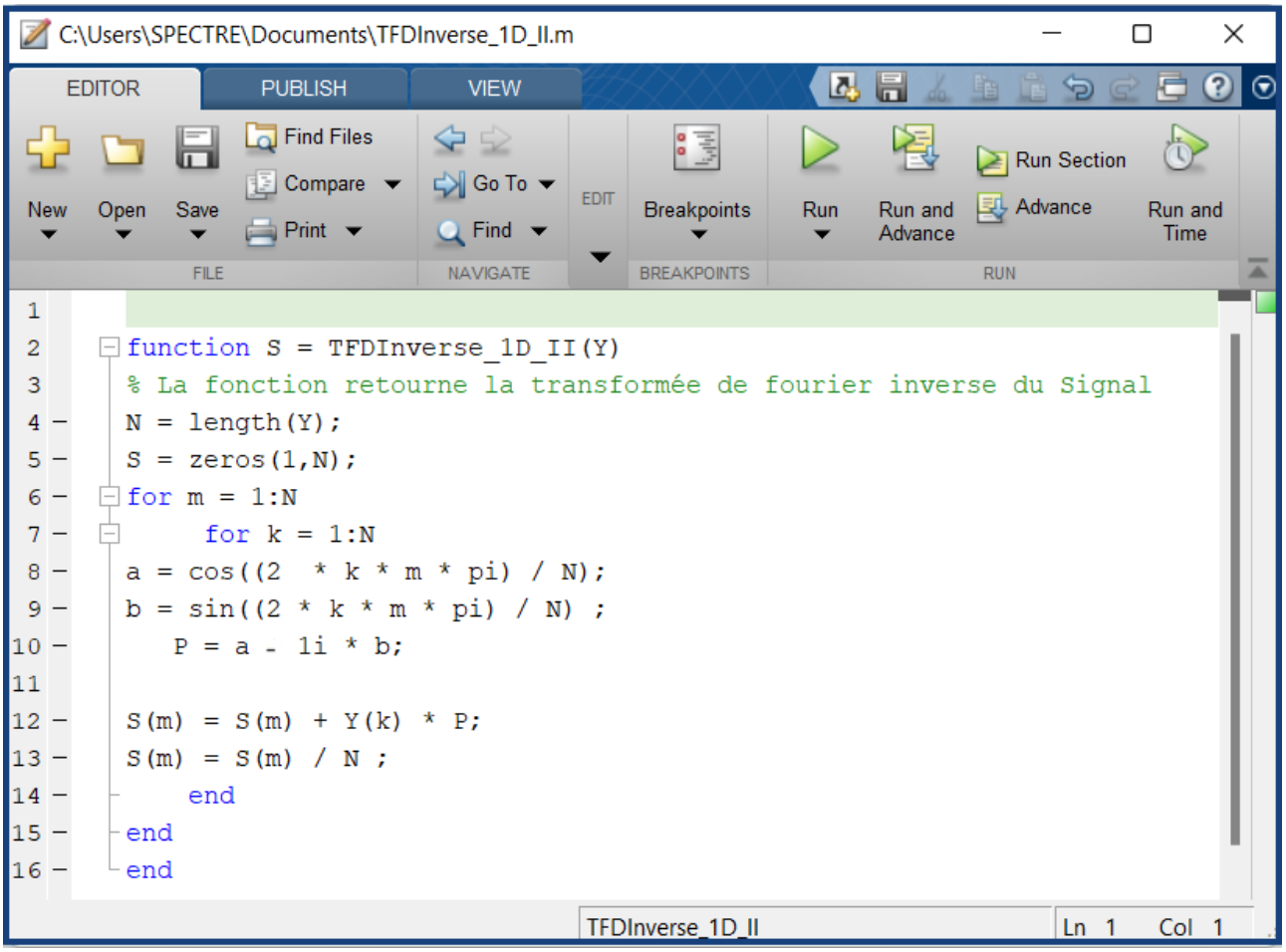
3.2 Transformée de Fourier inverse lente (1D) :

Code d'exécution :

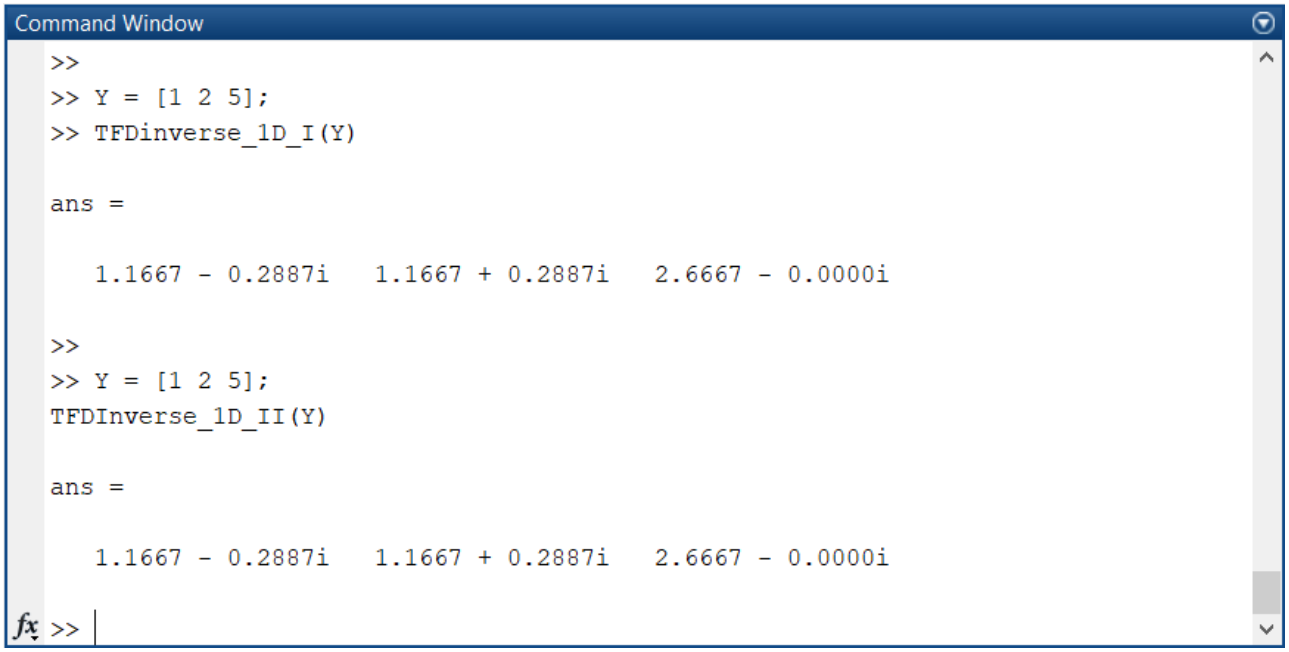
◇ **Méthode (1) :**



◇ **Méthode (2) :**



Exemple d'application :(les deux méthodes)



Exemple d'application : (TFD et ITFD)

```
Command Window

>> a = [1 2 5];
Y =TFD_1D(a)

Y =

   3.5000 + 0.8660i   3.5000 - 0.8660i   8.0000 + 0.0000i

>> TFDinverse_1D_I(Y)

ans =

   1.0000 - 0.0000i   2.0000 - 0.0000i   5.0000 + 0.0000i

fx >>
```

3.3 Transformée de Fourier lente (2D) :

Code d'exécution :

C:\Users\SPECTRE\Documents\TFD_2D34.m*

EDITOR | PUBLISH | VIEW

New | Open | Save | Find Files | Compare | Print | Go To | Find | Insert | Comment | Indent | Breakpoints | Run | Run and Advance | Run Section | Advance | Run and Time

FILE | NAVIGATE | EDIT | BREAKPOINTS

Save and run TFD_2D34 (F5)

```
1
2   function F = TFD_2D34(I)
3   %I est de dimension 2
4    [M,N]=size(I);
5   %retourne la dimension de la matrice (image) I
6   %Le nombre M de lignes et le nombre N de colonnes
7   F=zeros(M,N);
8   % retourne une matrice à M lignes et N colonnes contenant que des zéros
9   % TFD
10   for k=1:M
11      for l=1:N
12          for m=1:M
13              for n=1:N
14     a = cos(2*pi*(((k*m)/M)+((l*n)/N)));
15     b = sin(2*pi*(((k*m)/M)+((l*n)/N))) ;
16     w = a + 1i * b;
17
18              F(k,l)=F(k,l)+double(I(m,n)) * w;
19              end
20          end
21      end
22  end
23  end
```

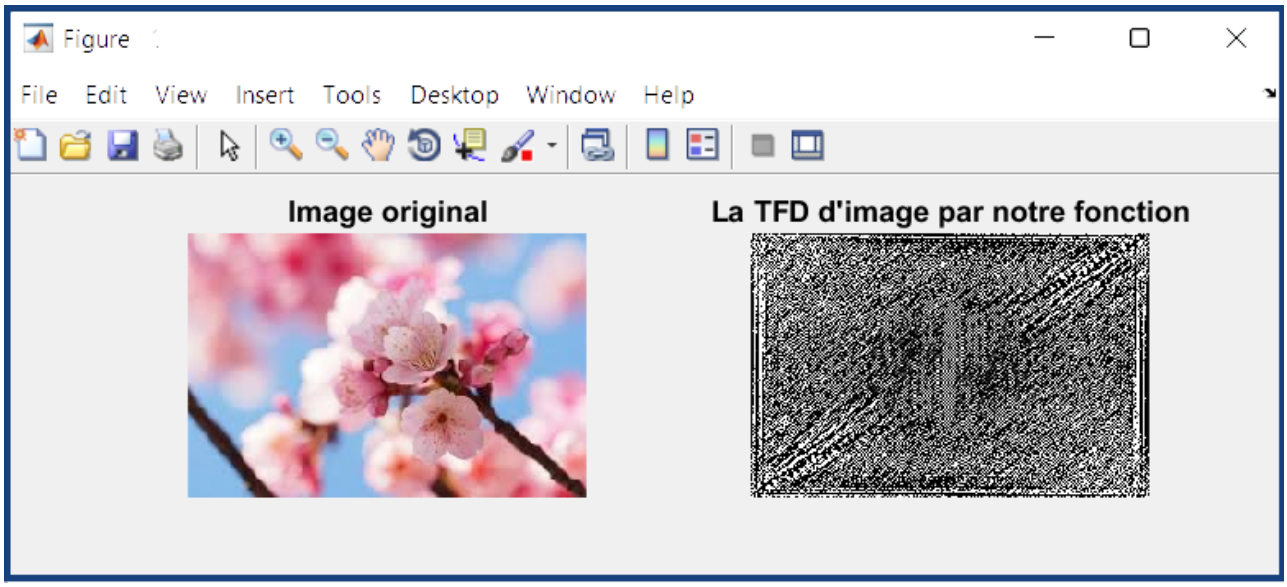
TFD_2D34 Ln 1 Col 1

Exemples d'application :

◇ **Exemple (1) :**

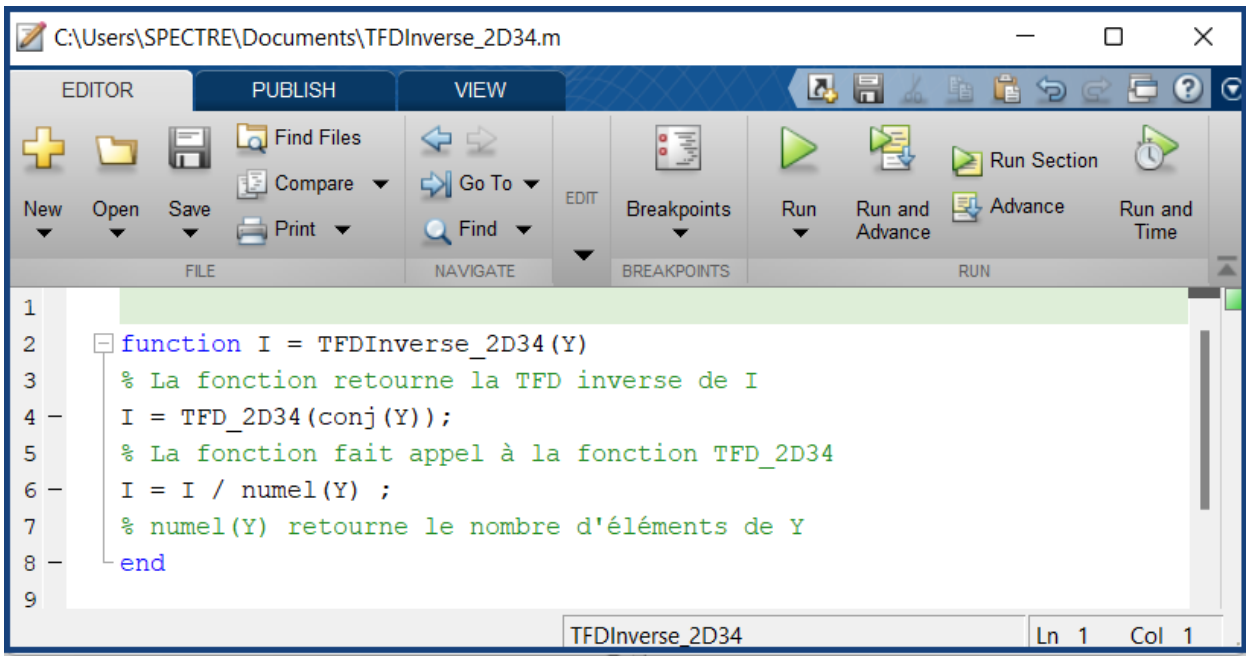
```
Command Window
>> a = [1 3 5; 2 4 6; 7 8 10]

a =

     1     3     5
     2     4     6
     7     8    10

>> TFD_2D34(a)

ans =

  -0.5000 + 0.8660i  -0.5000 - 0.8660i  14.5000 - 2.5981i
  -0.5000 + 0.8660i  -0.5000 - 0.8660i  14.5000 + 2.5981i
   8.5000 - 4.3301i   8.5000 + 4.3301i  46.0000 - 0.0000i
fx
```

◇ **Exemple (2) :**

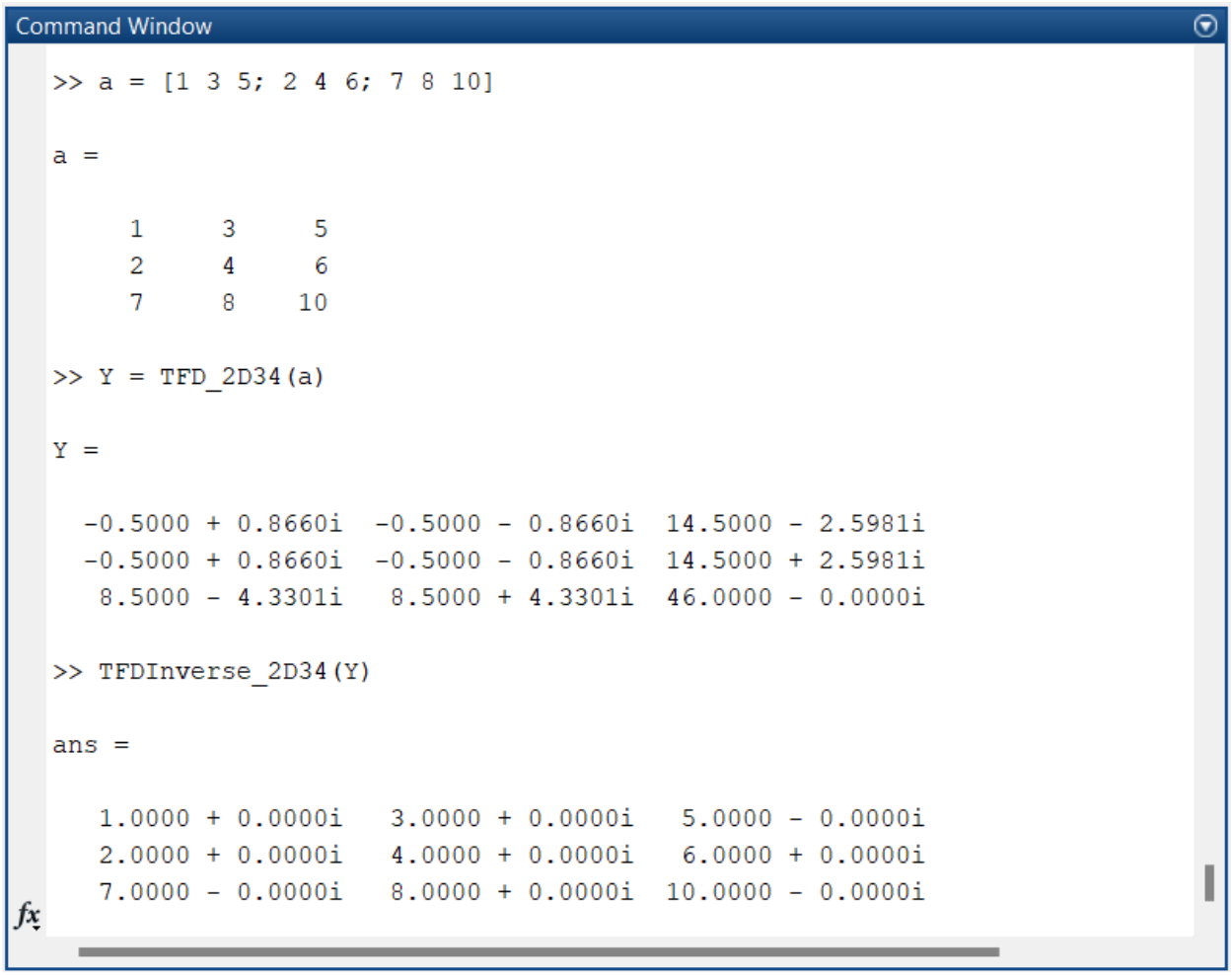


3.4 Transformée de Fourier inverse lente (2D) :

Code d'exécution :



Exemple d'application :(TFD2 et ITFD2)



3.5 Transformée de Fourier Rapide (1D) :

Code d'exécution :

◇ **Notre fonction :**

C:\Users\SPECTRE\Documents\FFT_1Ds.m

EDITOR | PUBLISH | VIEW

New | Open | Save | Find Files | Compare | Print | Go To | Find | Breakpoints | Run | Run and Advance | Run Section | Advance | Run and Time

FILE | NAVIGATE | EDIT | BREAKPOINTS | RUN

```
1
2   function Y = FFT_1Ds(S)
3   % S signal de dimension 1
4   % La fonction retourne la FFT du S
5   N = length(S);
6   % Le signal est de taille n = 2^k
7   if  N == 1
8
9          Y(1) = S(1);
10  else
11      k = (0:N/2-1);
12
13  Y_paire = FFT_1Ds(S(1:2:(N-1)));
14  % Y_paire contient les valeurs de S d'indices paires
15
16  Y_impaire = FFT_1Ds(S(2:2:N));
17  % Y_impaire contient les valeurs de S d'indices impaires
18
19  a = cos((2  * k * pi) / N);
20  b = sin((2 * k * pi) / N) ;
21  P = a - 1i * b;
22
23       Y1 = Y_paire + P.*Y_impaire;
24       Y2 = Y_paire - P.*Y_impaire;
25
26        Y = [Y1,Y2];
27  end
28  end
```

FFT_1Ds | Ln 1 Col 1

◇ **La fonction MATLAB :**

La transformée de Fourier rapide 1D est calculée en MATLAB par la fonction intégrée « fft ».

Exemples d'applications des deux fonctions

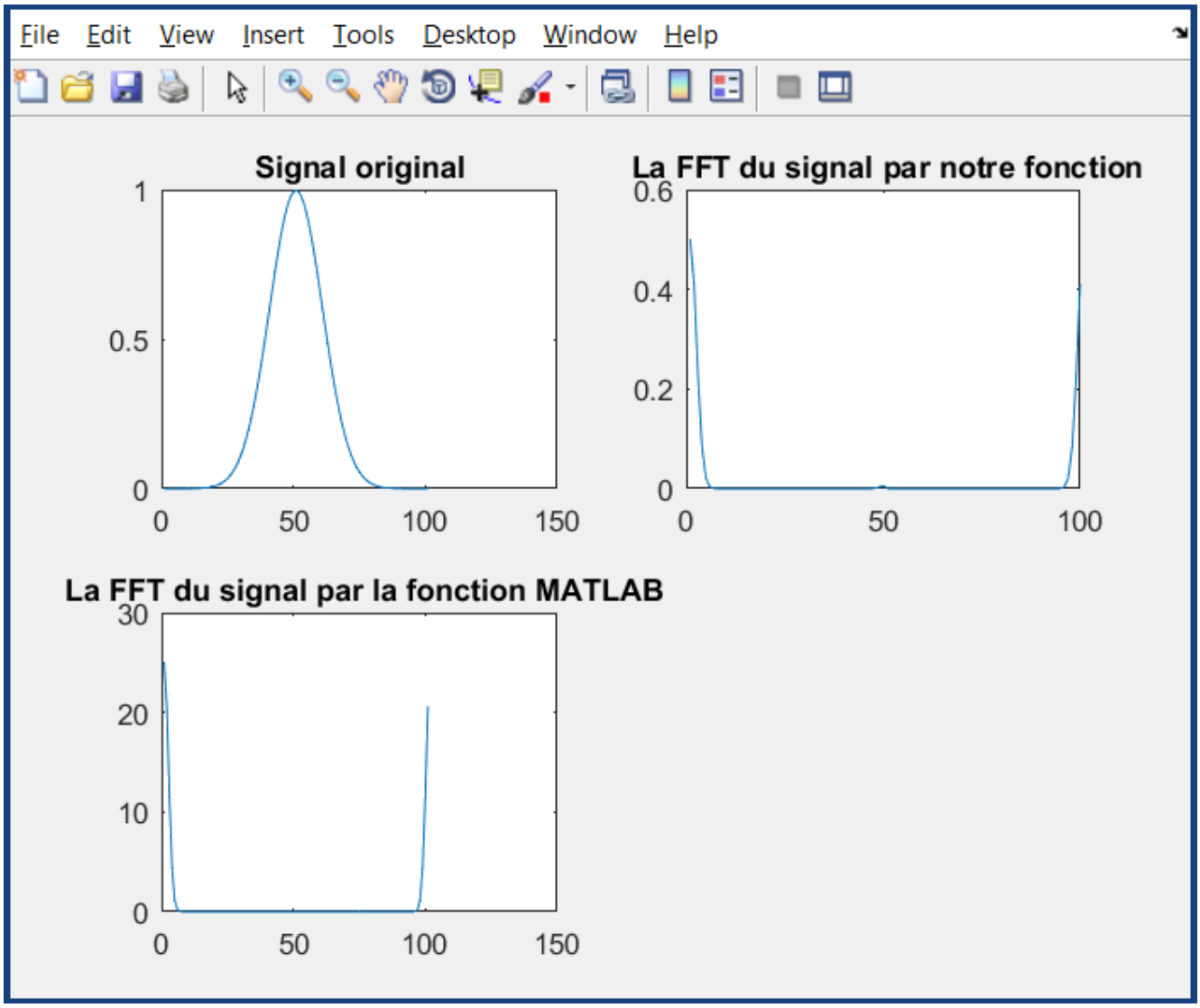
◇ **Exemple (1) :**

```
Command Window
>>
>> a=[1 2 5 7];
>> fft(a) %Fonction MATLAB

ans =

  15.0000 + 0.0000i  -4.0000 + 5.0000i  -3.0000 + 0.0000i  -4.0000 - 5.0000i

>> FFT_1Ds(a) % Notre fonction

ans =

  15.0000 + 0.0000i  -4.0000 + 5.0000i  -3.0000 + 0.0000i  -4.0000 - 5.0000i
```

◇ **Exemple (2) :**

```
Command Window
>>
>> t = -0.5:1/100:0.5;
signal = 1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
   A    = FFT_1Ds(signal);
% Par la fonction prédéfinie de MATLAB
   B    = fft(signal);
figure;
subplot(2,2,1), plot(signal); title("Signal original");
subplot(2,2,2), plot(abs(A)); title(" La FFT du signal par notre fonction");
subplot(2,2,3), plot(abs(B)); title(" La FFT du signal par la fonction MATLAB")
```

Notre signal 1D est réel et sa fft est complexe (la fonction abs() donne le module)


File Edit View Insert Tools Desktop Window Help
Signal original
1
0.5
0
0 50 100 150
La FFT du signal par notre fonction
0.6
0.4
0.2
0
0 50 100
La FFT du signal par la fonction MATLAB
30
20
10
0
0 50 100 150

3.6 Transformée de Fourier inverse (1D) :

Code d'exécution :

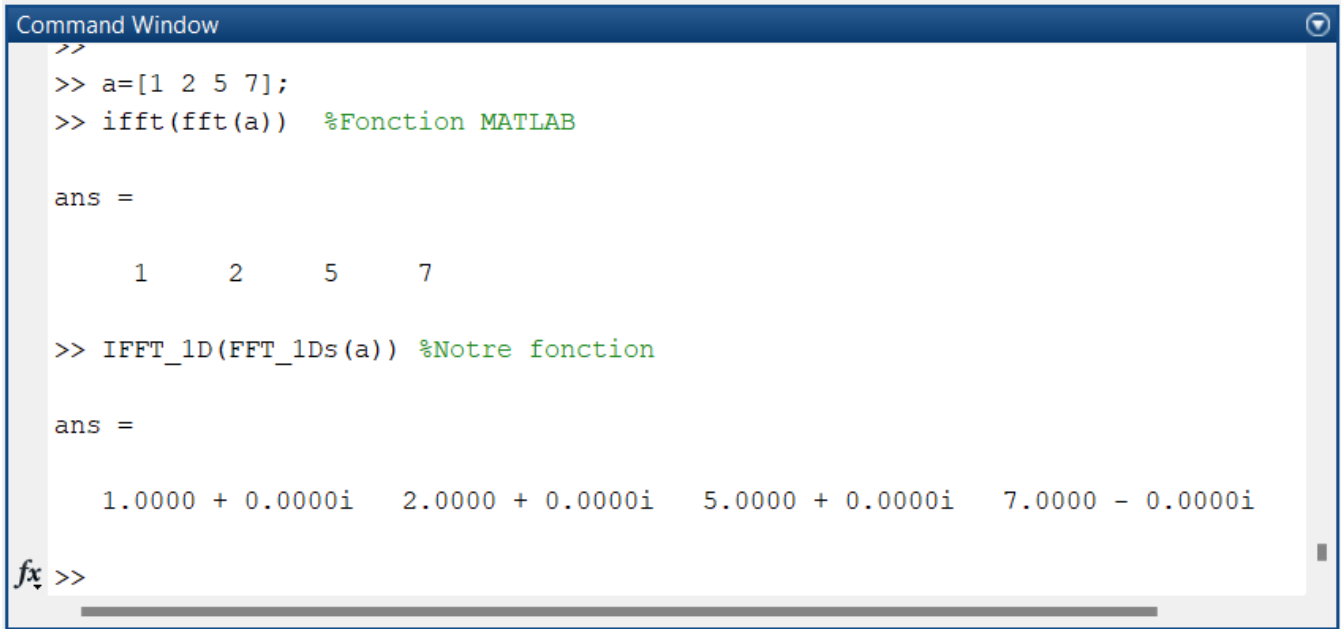
◇**Notre fonction :**

◇ **La fonction MATLAB :**

La transformée de Fourier rapide inverse 1D est calculée en MATLAB par la fonction intégrée « ifft ».

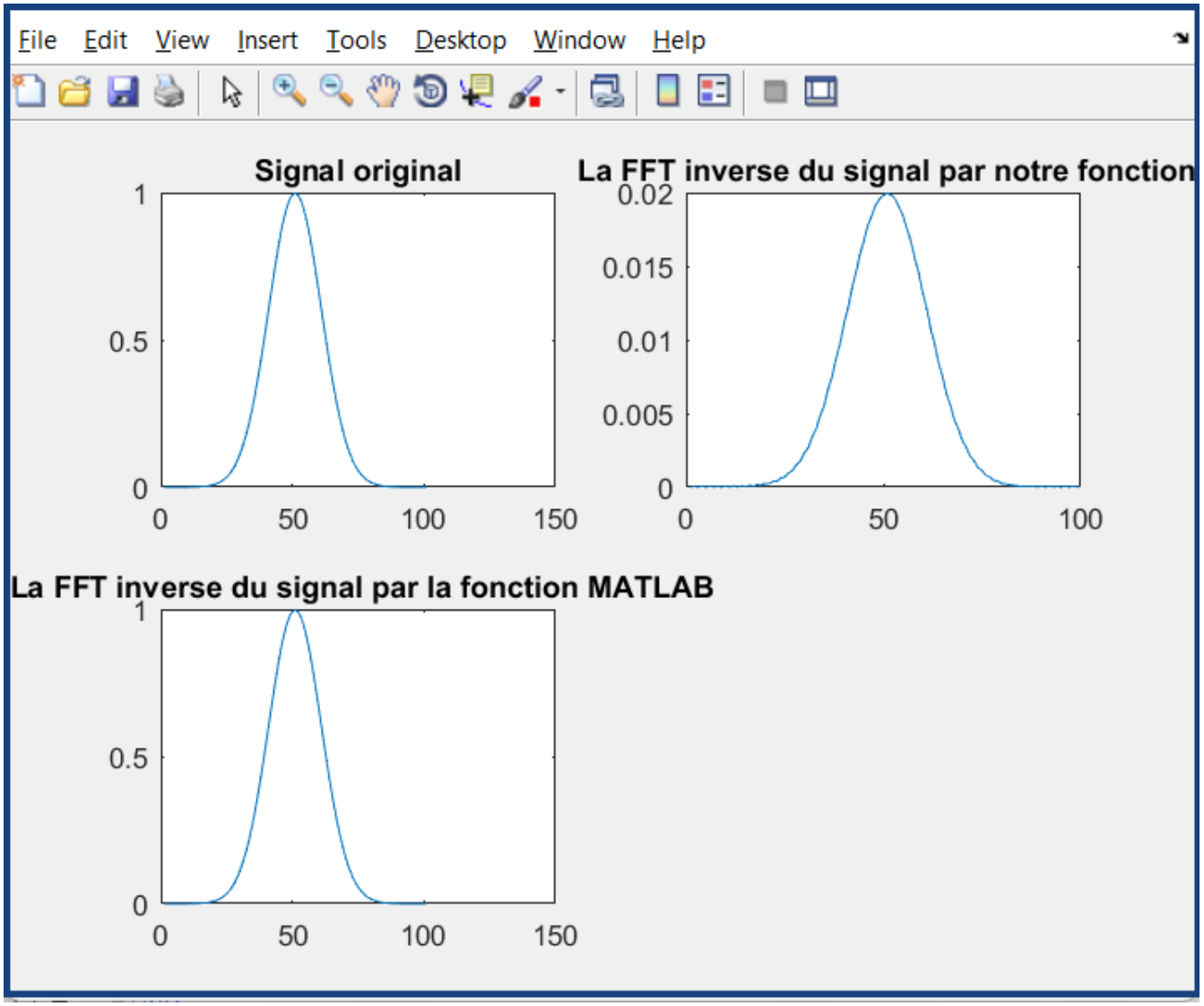
Exemples d'application :

- **Exemple (1) :**



- **Exemple (2) :**

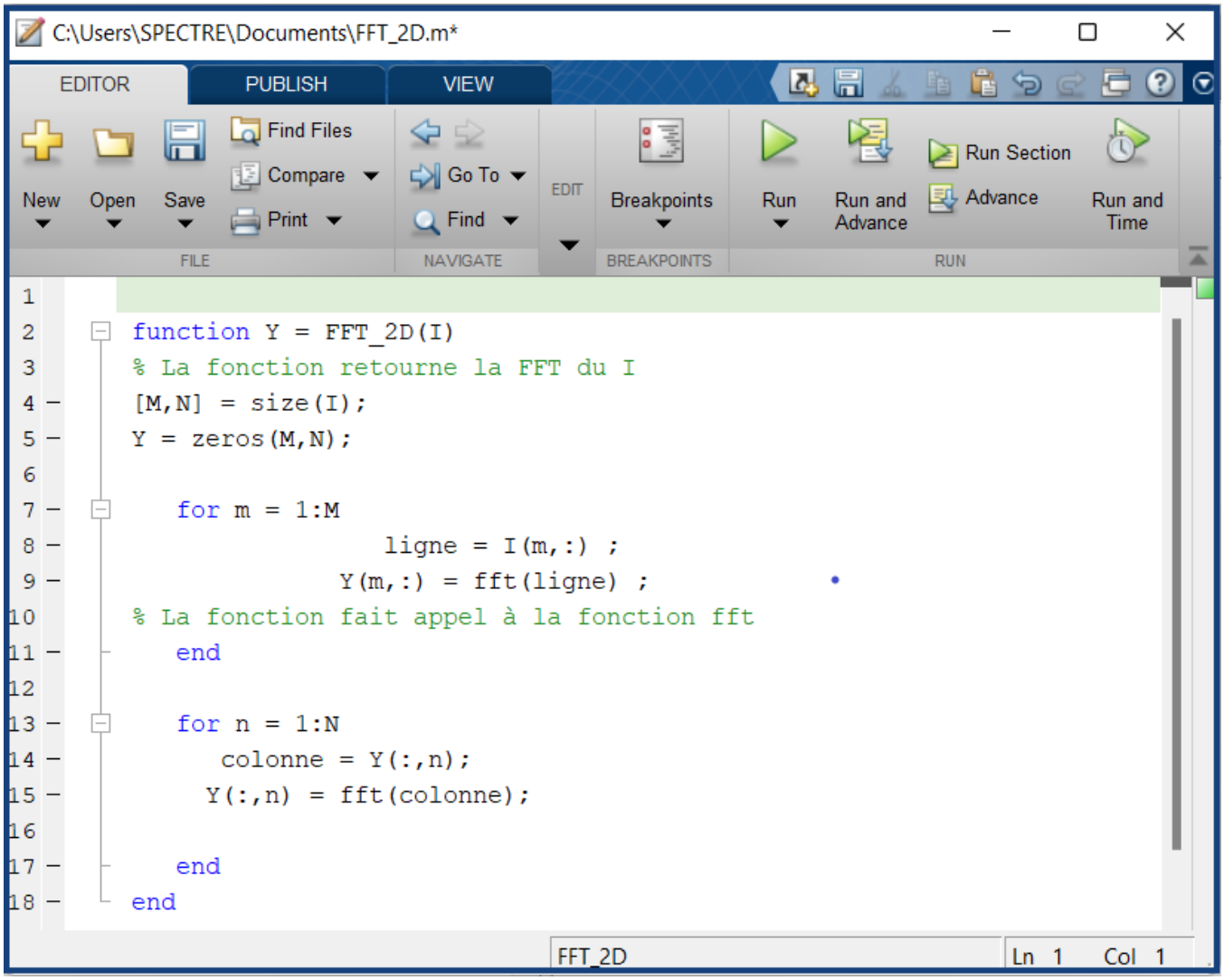
```
Command Window
>> t = -0.5:1/100:0.5;
signal = 1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));

   A    = FFT_1Ds(signal);
   C    = IFFT_1D(A);         % Notre fonction
   D    = ifft(fft(signal)); % La fonction prédéfinie de MATLAB
figure;
subplot(2,2,1), plot(signal); title("Signal original");
subplot(2,2,2), plot(abs(A)); title(" La FFT inverse du signal par notre fonction");
subplot(2,2,3), plot(abs(B)); title(" La FFT inverse du signal par la fonction MATLAB")
```



3.7 Transformée de Fourier Rapide (2D) :

Code d'exécution :

◇ **Notre fonction :**



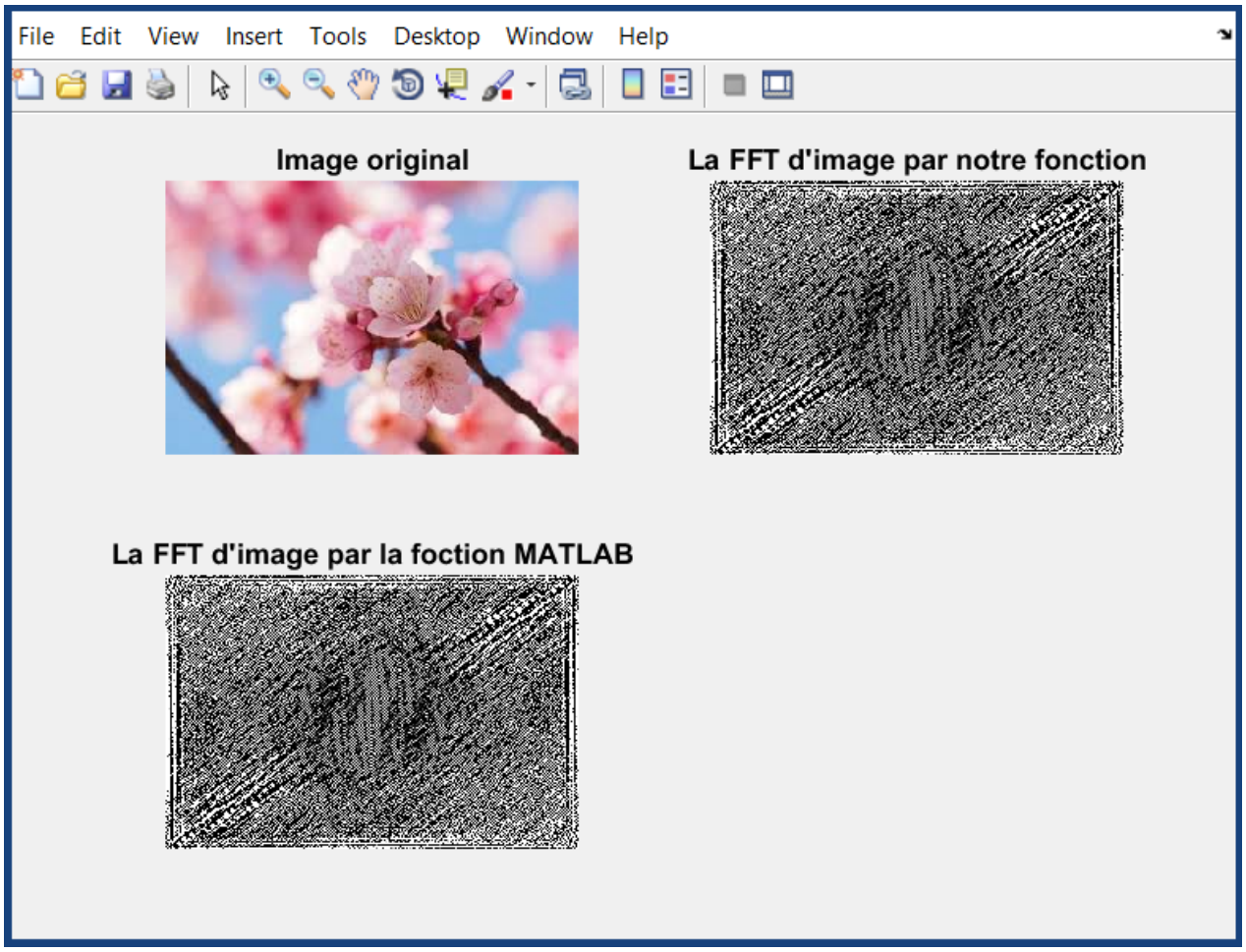
◇ **La fonction MATLAB :**

La transformée de Fourier rapide 2D est calculée en MATLAB par la fonction intégrée « fft2 ».

Exemples d'applications :

Command Window

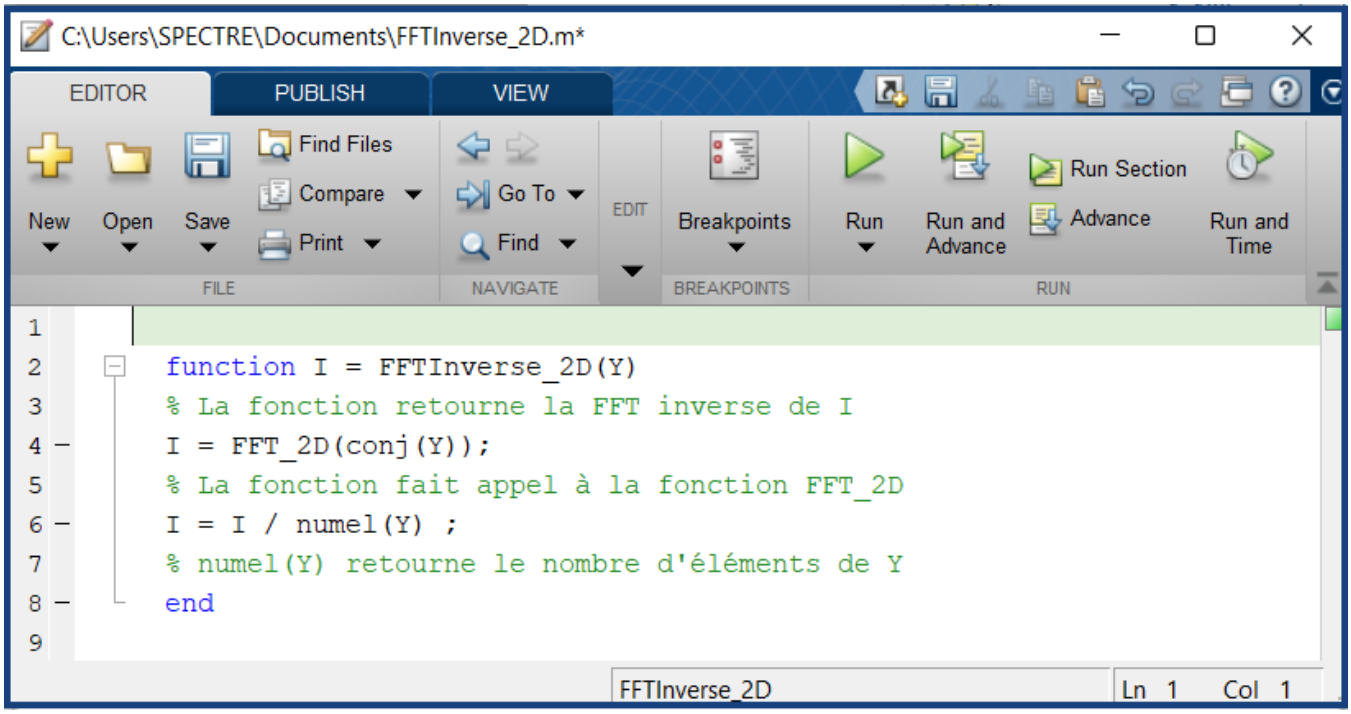
```
>> A = imread('imag.jpg');
I = rgb2gray(A);
I = im2double(I);
B = FFT_2D(I); % Notre fonction du FFT de I
C = fft2(I);   % La fonction MATLAB du FFT de I
figure;
subplot(2,2,1), imshow(A); title("Image original");
subplot(2,2,2), imshow(B); title("La FFT d'image par notre fonction");
subplot(2,2,3), imshow(C);title("La FFT d'image par la foction MATLAB")
```



3.8 Transformée de Fourier inverse (2D) :

Code d'exécution :

◇ **Notre fonction :**



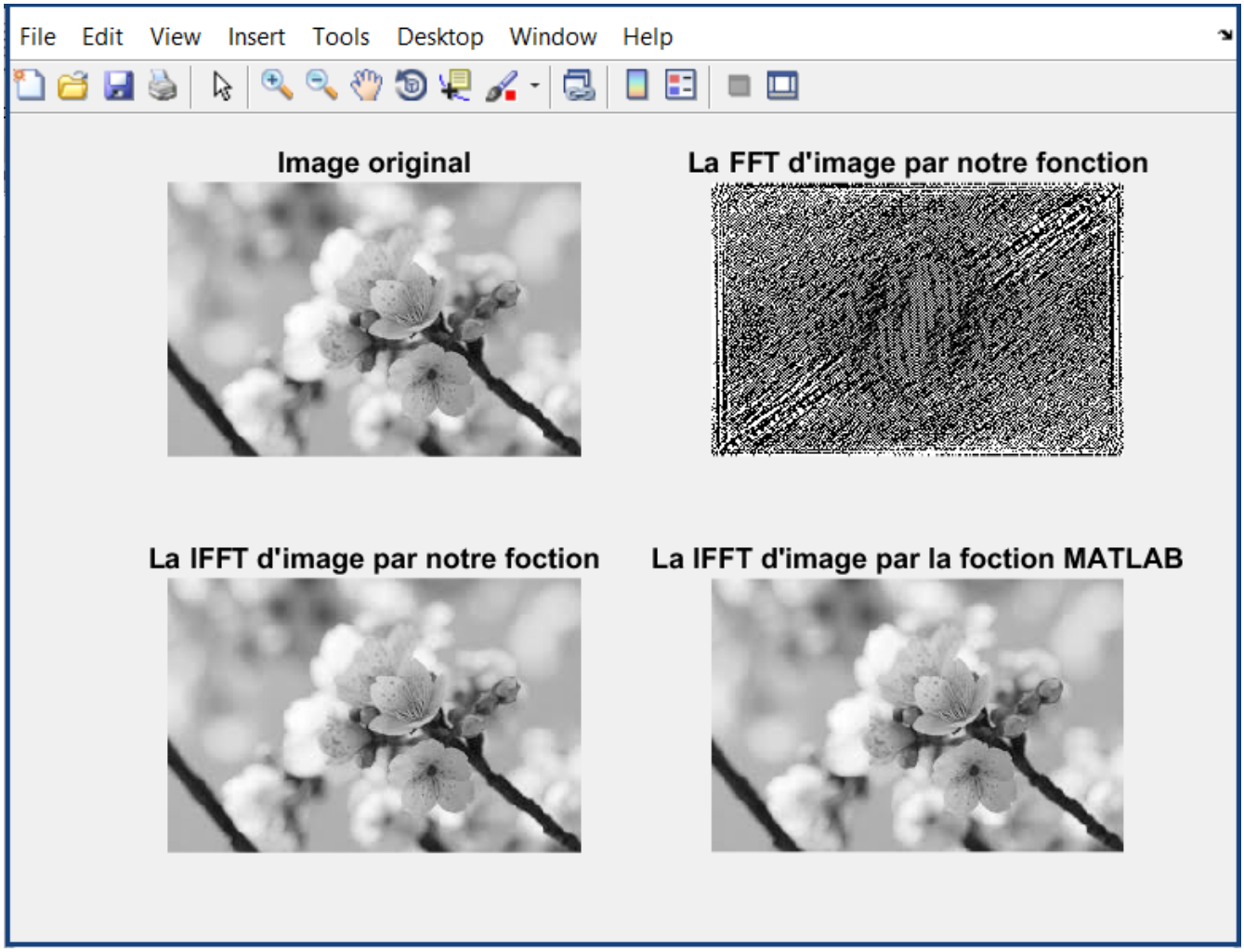
◇ **La fonction MATLAB :**

La transformée de Fourier rapide inverse 1D est calculée en MATLAB par la fonction intégrée « ifft2 ».

Exemple d'application :

Command Window

```
>> A = imread('imag.jpg');
I = rgb2gray(A);
I = im2double(I);
B = FFT_2D(I);
R = fft2(I);
C = FFTInverse_2D(B); % Notre fonction du FFT inverse
D = ifft2(R);         % La fonction MATLAB du FFT inverse
figure;
subplot(2,2,1), imshow(I); title("Image original");
subplot(2,2,2), imshow(B); title("La FFT d'image par notre fonction");
subplot(2,2,3), imshow(C);title("La IFFT d'image par notre foction");
subplot(2,2,4), imshow(D);title("La IFFT d'image par la foction MATLAB")
```



Conclusion

Dans ce travail, nous avons introduit les notions de base qui servent à la compréhension de différentes techniques de traitement d'images. Plusieurs méthodes classiques de traitement ont été proposés dans la littérature, nous avons présenté quelques-unes qui nous semblent les plus courantes dans le processus du traitement et d'analyse d'image qui permet de modifier le contenu des images afin de tirer l'information utile pour une application particulière.

Ce rapport a été dirigé vers l'un de ces techniques de traitement : la transformée de Fourier. Nous sommes arrivés à réaliser un programme dans le logiciel de haut niveau Matlab, qui transforme une image dans l'espace de Fourier et qui permet de diminuer le temps de la transformation.

De ce fait, nous avons utilisé des différents types de transformée de Fourier. Premièrement, nous avons décrit les notions de transformée de Fourier continue et ses propriétés. Deuxièmement, nous nous sommes intéressés aux éléments essentiels de la théorie de transformée de Fourier discret et Finalement, nous avons détaillé des algorithmes qui exploitent la formule de base pour le calcul de la Transformée de Fourier Discrète : $\hat{w}_{k,l} = \sum_{m,n=1}^{N} w_{m,n} e^{-2i\pi \frac{km+ln}{N}}$,
en utilisant l'implémentation naïve qui consomme beaucoup de temps de calcul et l'implémentation rapide (FFT). Et pour ce dernier, nous avons essayé de traiter en particulier l'algorithme de Cooley-Tuckey.

Table des figures

Bibliographie

Références

[1] D. Marr et E. Hildreth : “Theory of Edge Detection”, Proc. R. Soc. London, B 207, 187-217, 1980.

[2] B. Cipra, “The Best of the 20th Century : Editors Name Top 10 Algorithms”, SIAM News, vol. 33, no. 4, 2000.

[3] G. Peyré, “L’algèbre discrète de la transformée de Fourier”, Ellipse, 2004.

[4] J. W. Cooley et J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” Math. Comput., vol. 19, pp. 297–301, 1965.

[5] M. Mekideche, “Traitement d’images par des algorithmes basés sur le calcul fractionnaire”, Thése de doctorat, Université 20 Aout 1955 - Skikda, 2018.

[6] J. Monnier, “Transformée de Fourier”, INSA Toulouse, 2019.

[7] G. Baudoin et J.F. Bercher ,"Elements de Traitement du Signal" , Septembre 1998 - version 0.89.

[8] Guy Almouzni, “Traitement numérique des images” , EISTI, 2019-2020.

[9] Frédéric SUR, “Initiation au traitement du signal et applications”, École des Mines de Nancy, 2009-2012.

[10] C. Besse, “Signal, Fourier, Image”, L3 Mapi3, September 2020

[11] M. Van Droogenbroeck, “Acquisition et traitement de l’image, ULg, 2001.

[12] G. BAUDOIN et J.-F. BERCHER, M. Van Droogenbroeck, “Transformée de Fourier discrète”, École Supérieure d’Ingénieurs en Électrotechnique et Électronique, Novembre 2001 – version 0.1.

[13] F. Malgouyres, “Traitement d’images sur MATLAB”, MApI3.

[14] Francis Cottet, “Traitement des signaux et acquisition de données”, DUNOD, Paris, 2015.

Wibographie

[1̂] https ://interstices.info/de-la-transformee-de-fourier-a-l-imagerie-medicale/ .

[2̂] https ://www.geeksforgeeks.org/discrete-fourier-transform-and-its-inverse-using-matlab/ .

[3̂] https ://www.programmersought.com/article/68924054565/ .

[4̂] https ://glq2200.clberube.org/chapitres/docs/signal-fourier/ .

[5̂] https ://www.f-legrand.fr/scidoc/docimg/numerique/tfd/tfdimage/ .