UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

Rapport du projet :

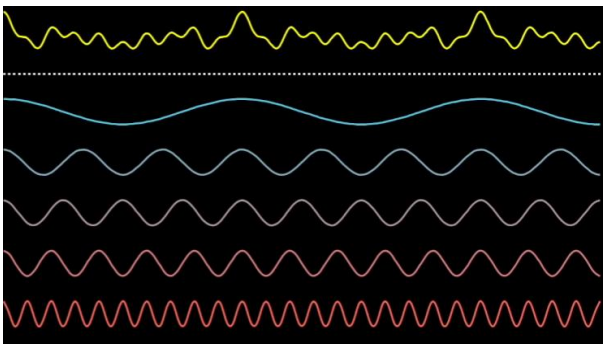
Outils Mathematiques pour l'informatique

Wassim Ennaji
Choukri Imane

Sommaire

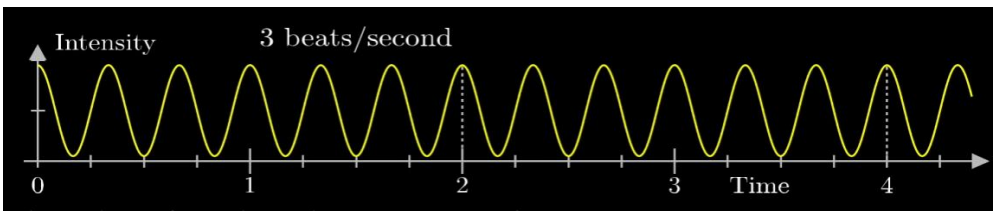
- Idée générale derrière la transformée de Fourier
- La transformée de Fourier directe discrète 1D et 2D
- La transformée de Fourier directe discrète inverse 1D et 2D
- La transformée de Fourier rapide discrète 1D et 2D
- La transformée de Fourier inverse rapide 1D et 2D

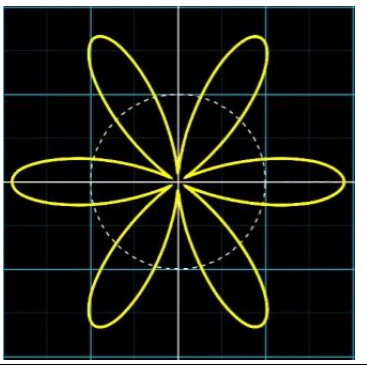
Idée générale derrière la transformée de Fourier



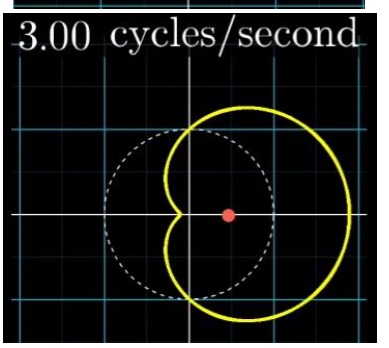
L'idée derrière la transformation de Fourier provient du fait que de nombreuses fonctions périodiques peuvent être décomposées en une somme de fonctions sinusoïdales (ondes sinusoïdales). En d'autres termes, la transformation de Fourier permet de passer d'une représentation temporelle d'un signal à une représentation fréquentielle, en indiquant quelles fréquences composent ce signal.

Pour visualiser ceci , on vient enrouler **un signal a une seule fréquence** (signal pur) autour d'un cercle , ici le signal a une fréquence de 3 battements par seconde , à laquelle on ajoute une autre fréquence qui caractérise le cercle sur lequel on a enroulé le signal (0.5 rotation par seconde .



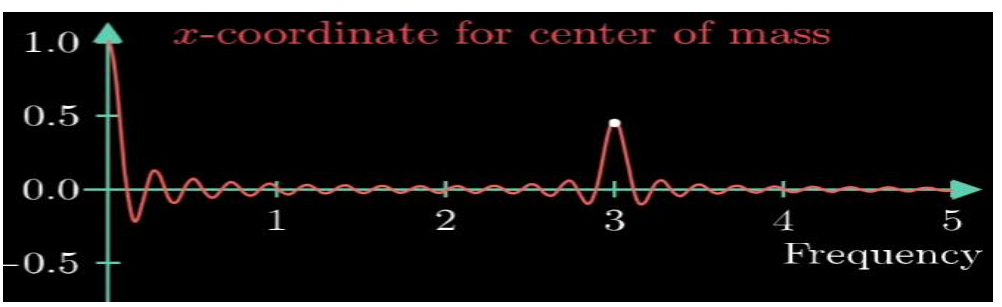


On peut ajuster la fréquence de l'enroulement du signal sur le cercle , de façon à ce qu'on l'enroule plus ou moins vite .



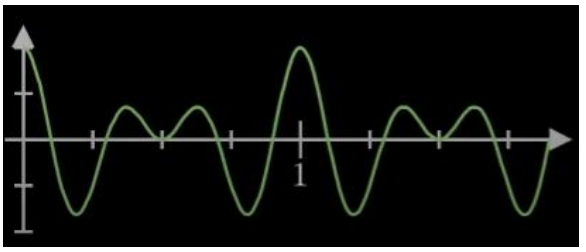
Supposant que ce graphe a une sorte de point de pesanteur , on remarque qu'en changeant la fréquence de l'enroulement autour du cercle , ce point reste proche du centre du cercle . Jusqu'à ce que cette fréquence soit égale à celle du signal , dans ce cas-là , le point passe à droite :

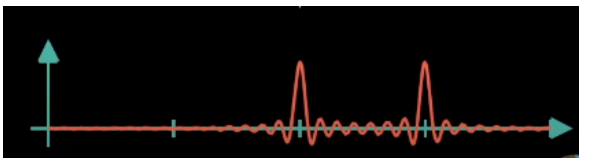
Pour visualiser le changement de position de ce point , en se focalise pour l'instant sur sa coordonnée x :



Cette méthode nous a permise donc de repérer la fréquence d'un signal donné .

Essayons maintenant , avec un signal composé de fréquences différentes :





On remarque alors que les piques sur le graphe résultant marquent les fréquences 2 et 3 dont ce graphe se compose .

Comment formuler cette idée ?

Il faut savoir que " le point de pesanteur " dont on parlait n'a pas qu'une seule coordonnée , mais deux . C'est donc un membre complexe qui a une valeur réelle et une imaginaire . **Pourquoi pas juste x et y ?** tout simplement car le domaine complexe est plus convenable quand on parle de rotation (sin et cos) .

- Pour représenter l'onde sinusoïdale qui varie en fonction du temps , on utilise la fonction exponentielle complexe :**e(−i2πt)** .
- En trigonométrie: ***e(−i2πt)* =cos(2π*t*)−*i*·sin(2π*t*)**

$$e^{-2\pi i f t}$$

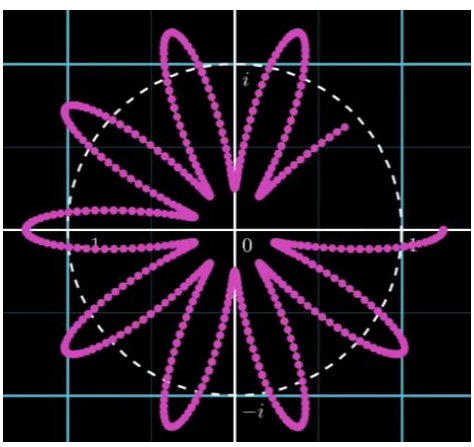
Cette formule représente la variation de phase de l'onde en fonction du temps, et l'exponentielle complexe permet de combiner amplitude et phase dans une représentation compacte : La partie réelle de e(−i2πt) représente l'amplitude du signal, et la partie imaginaire représente la phase.
Ensuite, on modifie la fréquence de 1 à f . On obtient :

- Soit g une fonction qui décrit l'intensité d'un signal en fonction du temps , en multipliant la formule ci-dessus et cette fonction , on obtient l'enroulement du signal autour du cercle en fonction des valeurs de cette fonction .

$$g(t)e^{-2\pi i f t}$$

Ceci représente l'intégralité du signal enroulé autour du cercle .
Mais comment se focaliser sur le point de pesanteur en question ?

- Pour cela , au lieu d'appliquer cette relation sur des points du graphe et ensuite diviser par le nombre de ces points , on prend directement une intégrale qui va définir cette relation sur TOUS les points de ce signal :



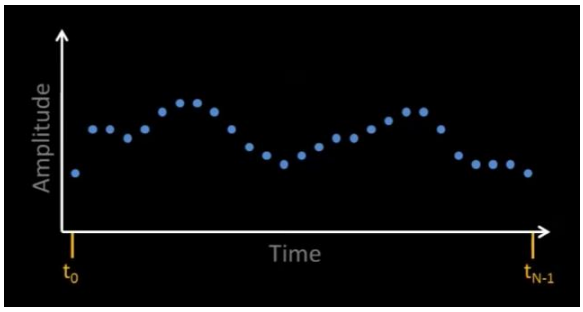
$$\hat{g}(f) = \int_{-\infty}^{+\infty} g(t)e^{-2\pi i f t}\,dt$$

On obtient ainsi , la formule de **la transformée de Fourier continue** (TFC) pour une fonction g(*t*) , elle donne une représentation de la fonction dans le domaine fréquentiel, montrant comment différentes fréquences contribuent au signal global.
La transformée de Fourier continue considère la fonction sur un domaine de temps qui s'étend de moins l'infini à plus l'infini. La notation $\int_{-\infty}^{\infty}$ dans la formule indique que l'intégration est effectuée sur toute la plage des valeurs possibles de la variable temporelle. Cela permet une représentation complète des fréquences présentes dans le signal, y compris les basses fréquences qui nécessiteraient une durée d'observation infinie pour être pleinement caractérisées.

La transformée de Fourier directe discrète 1D

La transition de la transformée de Fourier continue à la transformée de Fourier discrète implique généralement une discrétisation du temps et une approximation numérique.
Le passage de la TFC à la TFD implique essentiellement de remplacer l'intégration continue par une sommation discrète. Cela est dû au fait que, dans le domaine discret, nous n'avons plus une infinité continue d'instantanés, mais plutôt un ensemble fini d'échantillons.

Donc , contrairement à ce qu'on vu dans la transformée de fourrier continue , cette fois ci on traitera des signaux échantillonnés, c'est-à-dire des signaux qui sont mesurés à des intervalles réguliers.



Pour un signal échantillonné *x(n)*. La formule de la TFD est donnée par :

$$X(k) = \sum_{n=0}^{M-1} x(n)\, e^{-j2\pi kn/M}$$

- **x(n):** Il s'agit de la séquence que vous souhaitez transformer.
  Chaque *x(n)* représente la valeur de la séquence à un instant donné.
- **e(−j2πkn/M):** Ce terme correspond à une onde sinusoïdale discrète à la fréquence *k/M* . Il évolue en fonction de l'indice *n.*
- ***k*** *:* est l'indice de fréquence, prenant des valeurs de 0 à M−1.
- **M** : est le nombre total d'échantillons

Chaque terme exponentiel correspond à une fréquence spécifique, et le coefficient *X*[*k*] indique la contribution de cette fréquence au signal global.
L'exponentielle complexe ***e(−j2πkn/M)*** dans la formule représente une onde sinusoïdale à la fréquence normalisée k/M qui évolue en fonction de l'indice d'échantillon *n*.
La somme sur tous les échantillons combine ces termes sinusoïdaux pour représenter le signal dans le domaine fréquentiel.

Compléxité:

La complexitéest de l'ordre de *O*(*N*^2), où *N* est la longueur de la séquence. Cela signifie que le nombre total d'opérations élémentaires nécessaires pour la calculer augmente quadratiquement avec la longueur de la séquence.

L'implémentation sous Octave

```
function X = tf1d_D(x)
    N = length(x); % Longueur du signal
    X = zeros(1, N); % Initialisation du résultat de la transformée de Fourier

    for k = 1:N
        for n = 1:N
            X(k) = X(k) + x(n) * exp(-1i * 2 * pi * (n - 1) * (k - 1) / N);
        end
    end
end
```

La transformée de Fourier discrète inverse 1D

La transformée de Fourier discrète inverse est l'opération inverse de la transformée de Fourier discrète. Elle permet de reconstruire le signal original à partir de ses composantes fréquentielles obtenues par la TFD. La formule de la TFD inverse est donnée par :

$$x(n) = \frac{1}{M} \sum_{k=0}^{M-1} X(k) e^{j2\pi kn/M}$$

- $X[k]$ sont les composantes fréquentielles du signal dans le domaine de la fréquence.

L'IDFT donne une représentation du signal dans le domaine temporel en combinant différentes fréquences pondérées. Chaque terme $X[k]$ représente l'amplitude et la phase d'une composante sinusoïdale à la fréquence k. L'IDFT somme ces composantes pour reconstruire le signal dans le domaine temporel.Finalement , La somme est normalisée par la longueur du signal M.

L'implémentation sous Octave

```
function x_inv = ifft_1dVF(X)
    N = length(X); % Longueur du signal dans le domaine fréquentiel
    x_inv = zeros(1, N); % Initialisation du signal inverse

    for n = 1:N
        for k = 1:N
            x_inv(n) = x_inv(n) + X(k) * exp(1i * 2 * pi * (n - 1) * (k - 1) / N);
        end
        x_inv(n) = x_inv(n) / N; % Normalisation de l'IDFT
    end
end
```

La transformée de Fourier directe discrète 2D

La transformée de Fourier discrète bidimensionnelle est une extension de la transformée de Fourier discrète pour les signaux bidimensionnels, tels que les images. La 2D DFT permet de représenter un signal dans le domaine fréquentiel en analysant ses composantes dans les directions horizontale et verticale. Cette transformation est largement utilisée en traitement d'image, compression d'image, et d'autres domaines liés à l'analyse de données bidimensionnelles

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cdot e^{-j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

- $f[m,n]$ est la valeur du pixel à la position(m,n) dans l'image d'origine.
- $F(u,v)$ est le coefficient de la transformée de Fourier discrète bidimensionnelle à la fréquence (u,v) dans le domaine fréquentiel
- $e(-j2\pi(um/M + vn/M))$ sont des coefficients complexes qui correspondent aux fonctions sinusoïdales à différentes fréquences spatiales.
- Les termes u/M et v/N représentent les fréquences spatiales normalisées dans les directions horizontale et verticale, respectivement.
- u et v spécifient la fréquence spatiale dans chaque direction, m et n déterminent la position spatiale.

Etapes :

La 2D DFT est calculée en utilisant des sommes complexes similaires à la transformée de Fourier discrète unidimensionnelle , mais appliquées sur deux dimensions.
On peut interpréter le processus comme une séparation en deux étapes :

1. D'abord, on applique la transformée de Fourier discrète (DFT) en 1D le long des colonnes de l'image .
2. Puis on applique une DFT en 1D le long des lignes du résultat précédent.
3. Chaque terme $f[m,n] \cdot e^{-j2\pi(um/M + vn/N)}$ est sommé sur tous les pixels de l'image pour obtenir le coefficient $F(u,v)$.
4. Chaque coefficient F(u,v) représente l'amplitude et la phase d'une fréquence spatiale spécifique (u,v) dans l'image.
5. Finalelment , on répéte le processus pour toutes les valeurs de u et v pour obtenir la transformée de Fourier complète de l'image .

Compléxité:

La complexité est de l'ordre de $O(M2 \cdot N2)$, avec M et N étant les dimensions de l'image bidimensionnelle.

L'implémentation sous Octave

```
function [NI] = tfd2d(I)
    [hauteur, largeur] = size(I);
    NI = zeros(hauteur, largeur);

    for n = 1:hauteur
        for m = 1:largeur
            for l = 1:hauteur
                for k = 1:largeur
                    NI(n,m) = NI(n,m) + I(l,k) * exp(-2 * pi * 1i * (((n-1)*(l-1))/hauteur + ((m-1)*(k-1))/largeur));
                end
            end
        end
    end
end
```

La transformée de Fourier directe discrète 2D inverse

La 2D IDFT permet de reconstruire l'image spatiale à partir de ses composantes fréquentielles.

$$f(m,n) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \cdot e^{j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

- Le facteur 1/MN est utilisé pour normaliser l'inverse de la transformation , afin de s'assurer que l'amplitude totale de l'image reconstruite est proportionnelle à l'amplitude totale de l'image d'origine.
- Chaque terme dans la somme représente la contribution d'une fréquence spatiale (u,v) à la reconstruction de l'image.
- $f(m,n)$ représente la valeur de pixel dans l'image originale aux coordonnées (m,n) .

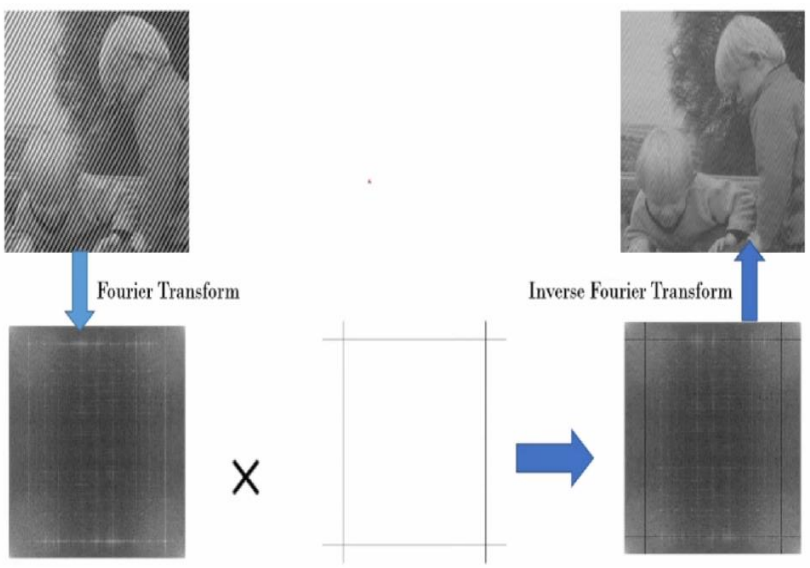
L'implémentation sous Octave

```
function [NI] = tfd2dI(I)
    [hauteur, largeur] = size(I);
    NI = zeros(hauteur, largeur);

    for n = 1:hauteur
        for m = 1:largeur
            for l = 1:hauteur
                for k = 1:largeur
                    NI(n,m) = NI(n,m) + I(l,k) * exp(-2 * pi * 1i * (((n-1)*(l-1))/hauteur + ((m-1)*(k-1))/largeur));
                end
            end
            NI(n,m)=abs(NI(n,m)*(1/(hauteur*largeur)));
        end
    end
end
```

Comment ça marche ?

Les fréquences spatiales élevées (**Composantes en Blanc)** représentent les variations rapides de l'intensité des pixels, tandis que les basses fréquences (**Composantes en Noir**) représentent les variations lentes.



- Appliquer la 2D DFT à l'image pour obtenir sa représentation dans le domaine fréquentiel.
- Le bruit peut apparaître comme des composantes de haute fréquence ou des pics indésirables dans le spectre fréquentiel.
- Créer un masque adapté qui servira à sélectionner ou atténuer certaines fréquences
- Multiplier la représentation dans le domaine fréquentiel de l'image par le masque qu'on a créé.
- Appliquer la 2D IDFT sur le produit résultant pour revenir à l'image spatiale modifiée

La transformée de Fourier rapide discrète 1D

L'idée fondamentale de la FFT est d'exploiter la régularité des calculs pour réduire la complexité algorithmique à *O*(*N*log*N*).
Cela est accompli en décomposant récursivement la DFT d'une séquence de longueur *N* en DFT de séquences de longueur moitié.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

Etapes :

- On commence par diviser la séquence en termes pair et impair (.
- On applique récursivement l'algorithme de Cooley-Tukey aux séquences pairs et impairs jusqu'à ce que la longueur de la séquence atteigne 2, pour laquelle la DFT peut être calculée directement.
- Finalement , en combines les résultats selon la relation suivante : $X[k]=Xp\ [k]+ e(-jN2\pi\ k) \cdot Xi\ [k]$.

La FFT réduit significativement la complexité temporelle par rapport à la méthode directe, ce qui est particulièrement crucial pour des séquences de données de grande taille.

Compléxité:

La complexité temporelle de l'algorithme de Cooley-Tukey est (*N*log*N*), ce qui est une amélioration significative par rapport à la méthode directe de *O*(*N*2). Cela rend la FFT particulièrement adaptée au traitement rapide de grandes quantités de données.

$$\sum_{n=0}^{N-1} a_n e^{-2\pi i n k/N}$$

paire / impaire — Changement de variable selon la parité

$$= \sum_{n=0}^{N/2-1} a_{2n} e^{-2\pi i (2n) k/N} + \sum_{n=0}^{N/2-1} a_{2n+1} e^{-2\pi i (2n+1) k/N}$$

$$= \sum_{n=0}^{N/2-1} a_n^{\text{paire}} e^{-2\pi i n k/(N/2)} + e^{-2\pi i k/N} \sum_{n=0}^{N/2-1} a_n^{\text{impaire}} e^{-2\pi i n k/(N/2)}$$

Transformée de fourrier discrète d'un tableau de taille $\frac{N}{2}$ d'indices paires + Transformée de fourrier discrète d'un tableau de taille $\frac{N}{2}$ d'indices impaires

On continue de subdiviser selon la parité jusqu'à ce qu'on atteigne les N transformées de fourrier discrète de taille 1 et de complexité ($\log_2 N$).
Donc, la complexité de cet algorithme est de:
$N \log_2(N)$.

L'implémentation sous Octave

```
function X = fft_1d(x)
    % Verification de la longueur du signal
    N = length(x);
    if N <= 1
        X = x;
        return;
    end
    % Division entre pair et impair
    pair = fft_1d(x(1:2:N-1));
    impair = fft_1d(x(2:2:N));
    % Facteur de rotation
    factor = exp(-2i * pi * (0:N/2-1) / N);
    % Combinaison des resultats
    X = zeros(1, N);
    X(1:N/2) = pair + factor .* impair;
    X(N/2+1:N) = pair - factor .* impair;
end
```

La transformée de Fourier rapide discrète 1D inverse

L'idée fondamentale derrière l'IFFT est similaire à celle de la FFT, mais elle consiste à inverser le processus. L'IFFT prend une séquence dans le domaine fréquentiel et la transforme en une séquence dans le domaine temporel.

Inversibilité : Si on applique la FFT à un signal, puis l'IFFT au résultat, on obtient le signal d'origine, à moins de petites erreurs numériques liées à la précision des calculs.

Elle est utilisée dans la reconstruction de signaux à partir de leur représentation fréquentielle.

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j\frac{2\pi}{N}kn}$$

- Cette formule inverse (identique à la transformée de fourrier discrète inverse 1D) prend la séquence $X[k]$ dans le domaine fréquentiel (résultat d'une FFT) et la transforme en la séquence $x[n]$ dans le domaine temporel.
- Chaque point de la séquence temporelle $x[n]$ est calculé en sommant les contributions fréquentielles pondérées provenant de toutes les fréquences k.

L'implémentation sous Octave

```
function x_inv = ifft_1d(X)
    % Nombre d'échantillons
    N = length(X);
    % Vérification de la condition d'arrêt pour la récursion
    if N <= 1
        x_inv = X;
        return;
    end
    % Division entre pair et impair (inversion)
    pair = ifft_1d(X(1:2:end));
    impair = ifft_1d(X(2:2:end));
    % Facteur de rotation (inversion)
    factor = exp(2i * pi * (0:N/2-1) / N);
    % Combinaison des résultats (inversion)
    x_inv = zeros(1, N);
    x_inv(1:N/2) = (pair + factor .* impair);
    x_inv(N/2+1:N) = (pair - factor .* impair);
end
```

La transformée de Fourier rapide discrète 2D

L'algorithme FFT 2D est une extension naturelle de l'algorithme FFT 1D, basé sur la décomposition en termes pair et impair. L'idée principale est d'appliquer la FFT 1D successivement dans les directions horizontale et verticale.

Etapes :

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

1. **Décomposition en Termes Pair et Impair dans la Direction Horizontale :**
   - La première étape consiste à décomposer chaque ligne de l'image en termes pair et impair. Cela s'effectue en appliquant la FFT 1D sur chaque ligne.
   - Les termes pairs sont les composantes correspondant aux indices pairs, et les termes impairs sont les composantes correspondant aux indices impairs.
2. **Décomposition en Termes Pair et Impair dans la Direction Verticale :**
   - La décomposition est ensuite appliquée dans la direction verticale sur chaque colonne résultante de l'étape précédente.
   - Encore une fois, les termes pairs et impairs sont identifiés dans chaque colonne.
3. **Combinaison des Résultats :**
   - À chaque niveau de la récursivité, les résultats des décompositions en termes pair et impair dans les deux directions sont combinés.
   - La combinaison est effectuée en utilisant des facteurs complexes liés aux indices fréquentiels, de manière similaire à la FFT 1D.
4. **Récursivité :**
   - Le processus de décomposition et de combinaison est répété de manière récursive jusqu'à ce que la taille des blocs atteigne 1x1 dans les deux dimensions. À ce stade, la FFT 2D devient équivalente à la FFT 1D.

Complexité:

Si *M* est la dimension en hauteur et *N* est la dimension en largeur de l'image, alors la complexité est de l'ordre de $O(M \cdot N \cdot \log2(M) \cdot \log2(N))$

L'implémentation sous Octave

```
function X = fft_2d(image)
    % Taille de l'image
    [M, N] = size(image);
    % Calcul de la FFT pour les lignes
    Y = zeros(M, N);
    for i = 1:M
        Y(i, :) = fft_1d(image(i, :));
    end
    % Calcul de la FFT pour les colonnes
    X = zeros(M, N);
    for j = 1:N
        X(:, j) = fft_1d(Y(:, j).'); % Transposée pour aligner les colonnes
    end
end
```

La transformée de Fourier rapide discrète 2D inverse

La FFT inverse 2D suit une approche similaire à la FFT 2D en termes de décomposition, récursivité et combinaison des résultats.
Elle permet de reconstruire une image bidimensionnelle à partir de sa représentation fréquentielle.

Etapes :

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \cdot e^{j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

- **FFT Inverse Verticale :**

Décomposition Verticale Inverse : On commence par inverser la décomposition verticale de la FFT 2D. Cela revient à appliquer la FFT inverse 1D verticalement à chaque colonne résultante.
FFT Inverse Verticale : Ensuite, la FFT inverse 1D est appliquée verticalement à chaque colonne pour reconstruire les séquences temporelles originales.

- **FFT Inverse Horizontale :**

Décomposition Horizontale Inverse : On inverse maintenant la décomposition horizontale de la FFT 2D. Cela revient à appliquer la FFT inverse 1D horizontalement à chaque ligne résultante.
FFT Inverse Horizontale : La FFT inverse 1D est appliquée horizontalement à chaque ligne pour reconstruire les séquences temporelles originales.

- Les résultats obtenus des FFT inverses horizontale et verticale sont combinés de manière appropriée pour reconstituer l'image bidimensionnelle d'origine.
- Les étapes sont répétées récursivement sur les sous-images jusqu'à ce que la taille des blocs atteigne 1x1.
- Comme pour la FFT 2D, la FFT inverse 2D nécessite une normalisation pour conserver l'amplitude correcte du signal.

L'implémentation sous Octave

```
function image_inv = ifft_2d(X)
    % Taille de l'image
    [M, N] = size(X);

    % Calcul de la TFI pour les lignes
    A = zeros(M, N);
    for i = 1:M
        A(i, :) = ifft_1d(X(i, :));
    end

    % Calcul de la TFI pour les colonnes
    image_inv = zeros(M, N);
    for j = 1:N
        image_inv(:, j) = ifft_1d(A(:, j).'); % Transposée pour aligner les colonnes
    end
end
```