OUTILS MATHÉMATIQUES POUR L'INFORMATIQUE

RAPPORT DE PROJET — CLUZEL OPHÉLIE & VAUBON SAMUEL

LA TRANSFORMÉE DE FOURIER

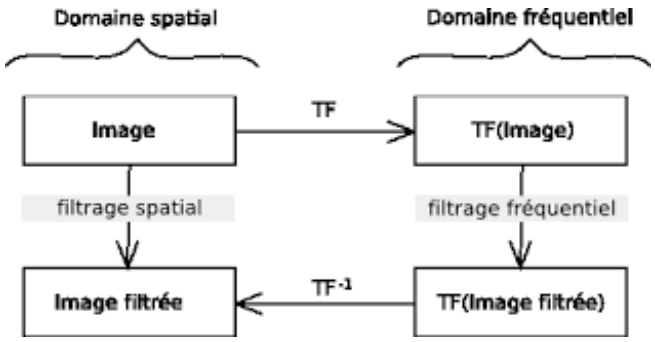
Introduction

QU'EST-CE QUE LA TRANSFORMÉE DE FOURIER ?

JOSEPH FOURIER, mathématicien français, affirma, dans un mémoire daté de 1807, qu'il était possible, dans certaines conditions, de décomposer une fonction périodique f sous la forme d'une somme infinie de signaux sinusoïdaux.

La transformée de Fourier est donc une opération qui permet de représenter en fréquence (développement sur une base d'exponentielles) des signaux qui ne sont pas périodiques. Il s'agit de l'analogue des séries de Fourier pour les fonctions périodiques (développement sur la base de fonctions sinusoïdales). Une fonction non périodique pouvant être considérée comme une fonction dont la période est infinie. Ce passage à la limite nous fait passer des séries aux intégrales.

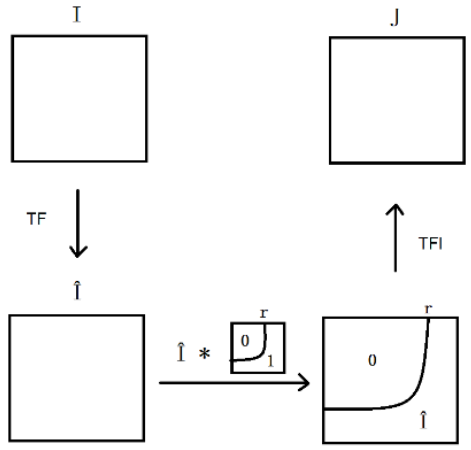
LA TRANSFORMÉE DE FOURIER ET LE TRAITEMENT D'IMAGES



On peut remarquer qu'il existe une dualité entre les domaines fréquentiel et spatial.

Le filtrage dans le domaine spatial équivaut à effectuer un produit de convolution entre un filtre (matrice 3x3 par exemple) et une Image. Malheureusement, en appliquant de gros masques de convolution, on peut constater que le temps de calcul devient très important. Pour pallier cela, on peut appliquer les masques dans l'espace des fonctions de Fourier qui rendront les calculs beaucoup plus rapides, car il existe un algorithme de calcul de la transformée très rapide. En effet le filtrage dans le domaine fréquentiel équivaut à une multiplication entre l'image et le filtre :

Schéma du filtrage dans le domaine fréquentiel

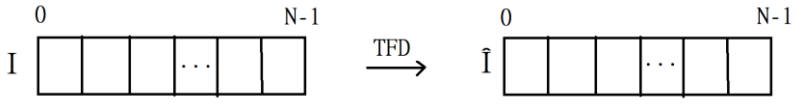


I. Transformée de Fourier directe discrète 1D et son inverse

Un signal 1D c'est en fait une image qui est égale à une matrice d'une ligne. Nous avons donc commencé par créer une matrice d'une ligne dans notre programme MATLAB :

```
image1D=[0 1 3 2 1 0 5 0];
```

Si notre image a une taille de N cases sa transformée sera de même taille.



Dans notre fonction on commence donc par déclarer une nouvelle matrice qui fera la même taille que l'image de base :

```
function[image1DTF] = TFD1D(image1D)
[M,N] = size(image1D);
image1DTF=zeros(1, N);
```

La formule mathématique de la transformée de Fourier directe discrète 1D d'une image de taille N est la suivante :

$$\hat{I}(u) = \sum_{x=0}^{N-1} I(x) e^{(-\frac{2i\pi ux}{N})}$$

On l'adapte donc à notre programme MATLAB :

```
for u=1:N
    for x=1:N
        image1DTF(u) = image1DTF(u) + image1D(x)*exp((-2*1i*pi*u*x)/N);
    end
end
```

Pour la transformée de Fourier directe discrète 1D inverse c'est le même principe, il y a juste un signe qui change dans la formule :

$$\hat{I}(u) = \sum_{x=0}^{N-1} I(x) e^{(\frac{2i\pi ux}{N})}$$

On l'adapte donc à notre programme MATLAB :

```
function[image1DTF] = TIFD1D(image1D)
[M,N] = size(image1D);
image1DTF=zeros(1, N);
for u=1:N
    for x=1:N
        image1DTF(u) = image1DTF(u) + image1D(x)*exp((2*1i*pi*u*x)/N);
    end

end
end
```

En appliquant nos fonctions à notre image 1D cela nous donne :

```
image1DTF=TFD1D(image1D);
image1DTFI=TIFD1D(image1DTF);
disp(image1D);
disp(image1DTF);
disp(image1DTFI);
```

```
   0     1     3     2     1     0     5     0

-1.2929 + 1.1213i   1.0000 + 7.0000i  -2.7071 + 3.1213i  -6.0000 - 0.0000i  -2.7071 - 3.1213i   1.0000 - 7.0000i  -1.29

 0.0000 + 0.0000i   8.0000 - 0.0000i  24.0000 - 0.0000i  16.0000 - 0.0000i   8.0000 + 0.0000i  -0.0000 + 0.0000i  40.00
```

On remarque que l'on ne retombe pas sur les valeurs de l'image d'origine. Nous devons donc calculer le coefficient de passage de l'image transformée à l'image de base. Pour cela il nous

suffit de diviser la valeur d'arrivée par la valeur de départ. On remarque que le coefficient est égal à 8 donc à N.

Il ne nous reste plus qu'à l'ajouter dans notre fonction inverse en divisant chaque case par celui-ci :

```
function[image1DTF] = TIFD1D(image1D)
[M,N] = size(image1D);
image1DTF=zeros(1, N);
for u=1:N
    for x=1:N
        image1DTF(u) = image1DTF(u) + image1D(x)*exp((2*1i*pi*u*x)/N);
    end
    image1DTF(u)=image1DTF(u)/N;
end
end
```

Et maintenant si on lance à nouveau nos fonctions cela nous donne :

```
    0     1     3     2     1     0     5     0
-1.2929 + 1.1213i   1.0000 + 7.0000i  -2.7071 + 3.1213i  -6.0000 - 0.0000i  -2.7071 - 3.1213i   1.0000 - 7.0000i  -1.2929 - 1.1213i  12.0000 + 0.000
 0.0000 + 0.0000i   1.0000 - 0.0000i   3.0000 - 0.0000i   2.0000 - 0.0000i   1.0000 + 0.0000i  -0.0000 + 0.0000i   5.0000 + 0.0000i   0.0000 + 0.000
```

Cette fois-ci on retombe bien sur l'image de départ.

Complexité : O(N^2) : Par pixel on a N calcul (somme de taille N) fois N le nombre total de pixels de l'image : N*N= N^2.

II. Transformée de Fourier directe discrète 2D et son inverse

Un signal 2D c'est en fait une image qui est égale à une matrice à deux dimensions : lignes et colonnes. Nous avons donc commencé par créer une matrice à deux dimensions dans notre programme MATLAB (ici c'est une matrice carrée de taille 6x6) :

```
image2D=[0 1 5 4 8 7 ;  4 2 9 8 5 1 ;  5 2 4 6 2 1 ;  1 2 5 4 3 6 ;  2 5 4 1 2 5 ;  0 3 2 1 0 5 ];
```

La formule mathématique de la transformée de Fourier directe discrète 2D d'une image de taille M*N est la suivante :

$$\hat{I}(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)e^{(-2i\pi(\frac{ux}{M}+\frac{vy}{N}))}$$

On l'adapte donc à notre programme MATLAB :

```
function[image] = TFD2D(image2D)
[M,N] = size(image2D);
image=zeros(M, N);
for u=1:N
    for v=1:M
        for x=1:N
            for y=1:M
                image(u,v) = image(u,v) + image2D(x,y)*exp((-2*1i*pi*(((u*x)/N)+((v*y)/M))));
            end
        end
    end
end
```

Pour la transformée de Fourier directe discrète 2D inverse c'est le même principe, il y a juste un signe qui change dans la formule :

$$\hat{I}(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)e^{(2i\pi(\frac{ux}{M}+\frac{vy}{N}))}$$

On l'adapte donc à notre programme MATLAB :

```
function[image] = TIFD2D(image2D)
[M,N] = size(image2D);
image=zeros(M, N);
for u=1:N
    for v=1:M
        for x=1:N
            for y=1:M
                image(u,v) = image(u,v) + image2D(x,y)*exp((2*1i*pi*(((u*x)/N)+((v*y)/M))));
            end
        end
    end
end
```

En appliquant nos fonctions à notre image 2D cela nous donne (en voici un extrait) :

```
image2DTF=TFD2D(image2D);
image2DTFI=TIFD2D(image2DTF);
disp(image2D);
disp(image2DTFI);
```

```
  0     1     5     4     8     7
  4     2     9     8     5     1
  5     2     4     6     2     1
  1     2     5     4     3     6
  2     5     4     1     2     5
  0     3     2     1     0     5

 1.0e+02 *

 0.0000 + 0.0000i   0.3600 + 0.0000i   1.8000 + 0.0000i   1.4400 + 0.0000i   2.8800 - 0.0000i
 1.4400 + 0.0000i   0.7200 - 0.0000i   3.2400 - 0.0000i   2.8800 + 0.0000i   1.8000 - 0.0000i
```

On remarque que l'on ne retombe pas sur les valeurs de base de l'image. On calcule donc le coefficient qui est égal à 0,36 donc il est égal à (N*M/100). Maintenant si on lance à nouveau

nos fonctions cela nous donne :

```
     0     1     5     4     8     7
     4     2     9     8     5     1
     5     2     4     6     2     1
     1     2     5     4     3     6
     2     5     4     1     2     5
     0     3     2     1     0     5

1.0e+02 *

   0.0000 + 0.0000i   1.0000 + 0.0000i   5.0000 + 0.0000i   4.0000 + 0.0000i   8.0000 - 0.0000i
   4.0000 + 0.0000i   2.0000 - 0.0000i   9.0000 - 0.0000i   8.0000 + 0.0000i   5.0000 - 0.0000i
```

Cette fois-ci on retombe bien sur les mêmes valeurs.

Complexité : $O(N^2M^2) = O(N^4)$ La transformée de Fourier discrète 2D est en fait la composée de deux transformées de Fourier 1D appliquées aux lignes et aux colonnes. On va donc effectuer N^2 fois N^2 calculs = N^4. Le temps de calcul est ici très important même si l'on ne s'en rend pas forcément compte car nous travaillons avec des images de très petite taille, mais appliqué à des images plus grandes, l'efficacité de notre fonction serait compromise.

III. Transformée de Fourier rapide discrète 1D et son inverse

Pour effectuer la transformée de Fourier rapide discrète on va reprendre la formule de la transformée de Fourier directe discrète 1D.

$$\hat{I}(u) = \sum_{x=0}^{N-1} I(x) e^{(-\frac{2i\pi ux}{N})}$$

On va la décomposer en deux parties : les valeurs à indice pairs et les valeurs à indices impairs.

$$\hat{I}(u) = \sum_{\substack{x=0 \\ (x\ pair)}}^{N-1} I(x) e^{(-\frac{2i\pi ux}{N})} + \sum_{\substack{x=0 \\ (x\ impair)}}^{N-1} I(x) e^{(-\frac{2i\pi ux}{N})}$$

On va supposer que l'on travaille sur des matrices de la taille d'une puissance de 2 (taille en général des matrices). On pose donc : $N = 2^n$. En partant de ce principe on peut simplifier la formule :

$$\hat{I}(u) = \sum_{\substack{x=0 \\ (x\ pair)}}^{N-2} I(x) e^{(-\frac{2i\pi ux}{N})} + \sum_{\substack{x=1 \\ (x\ impair)}}^{N-1} I(x) e^{(-\frac{2i\pi ux}{N})}$$

Maintenant on va essayer de simplifier la première partie de cette addition : le cas des valeurs à indices pairs. On pose donc x=2x'(nombre pair).

$$\hat{I}(u) = \sum_{x'=0}^{\frac{N}{2}-1} I(2x')e^{(-\frac{2i\pi u 2x'}{N})} + \sum_{\substack{x=1 \\ (x\ impair)}}^{N-1} I(x)e^{(-\frac{2i\pi ux}{N})}$$

Comme on a $\frac{\frac{a}{b}}{\frac{c}{d}} = \frac{ad}{bc}$ on peut simplifier :

$$\hat{I}(u) = \sum_{x=0}^{\frac{N}{2}-1} I(2x)e^{(-\frac{2i\pi ux}{\frac{N}{2}})} + \sum_{\substack{x=1 \\ (x\ impair)}}^{N-1} I(x)e^{(-\frac{2i\pi ux}{N})}$$

Maintenant passons à la simplification de la deuxième partie de l'addition : le cas des valeurs à indices impairs. On pose donc x=2x'+1(nombre impair).

$$\hat{I}(u) = \sum_{x=0}^{\frac{N}{2}-1} I(2x)e^{(-\frac{2i\pi ux}{\frac{N}{2}})} + \sum_{x'=0}^{\frac{N}{2}-1} I(2x'+1)e^{(-\frac{2i\pi u(2x'+1)}{N})}$$

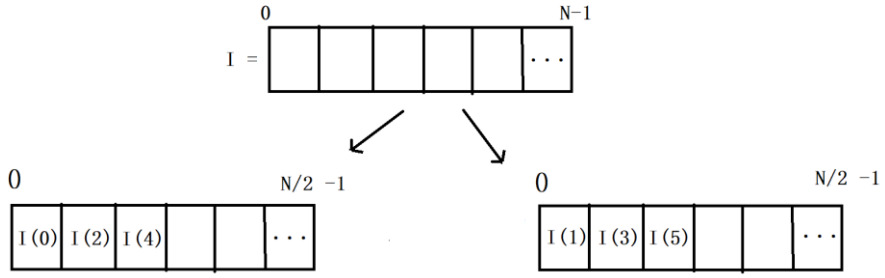
Comme on a $e^{(a+b)} = e^a * e^b$, on peut simplifier :

$$\hat{I}(u) = \sum_{x=0}^{\frac{N}{2}-1} I(2x)e^{(-\frac{2i\pi ux}{\frac{N}{2}})} + \sum_{x=0}^{\frac{N}{2}-1} I(2x+1)e^{(-\frac{2i\pi u}{N})} * e^{(-\frac{2i\pi ux}{\frac{N}{2}})}$$

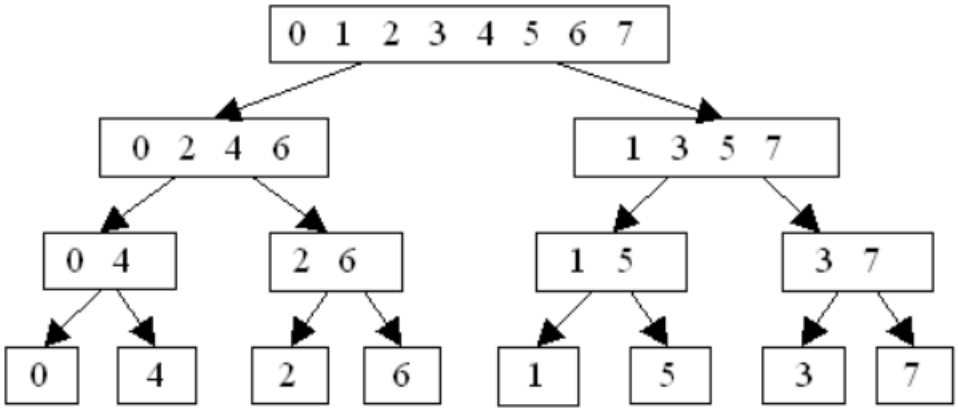
Comme la première exponentielle ne dépends pas de x on peut la sortir de la somme, on obtient donc finalement :

$$\hat{I}(u) = \sum_{x=0}^{\frac{N}{2}-1} I(2x)e^{(-\frac{2i\pi ux}{\frac{N}{2}})} + e^{(-\frac{2i\pi u}{N})} \sum_{x=0}^{\frac{N}{2}-1} I(2x+1)e^{(-\frac{2i\pi ux}{\frac{N}{2}})}$$

Ainsi on a décomposé notre image de taille N en deux images de taille $\frac{N}{2}$.



La transformée de Fourier rapide 1D sera donc un appel récursif de transformées de Fourier jusqu'à ce qu'on arrive à des matrices à un seul indice :



Basé sur ce principe voici notre programme MATLAB :

```
function[image] =TFR1D(image1D)
[M,N] = size(image1D);
image=zeros(1, N);

if N==1
   image=image1D/N;
else

Pair = zeros(1, N / 2);
Impair = zeros(1, N / 2);
for u = 1 : N
 if mod(u, 2) == 0
 Pair(u / 2) = image1D(u);
 else
 Impair( ( u + 1 ) / 2 ) = image1D(u);
 end
end

FTPair=TFR1D(Pair);
FTImpair=TFR1D(Impair);
for u=1:N/2
    coef = exp( (- 2 * i * pi * u ) / N );
    image(u)=FTPair(u)+(coef * FTImpair(u));
    image((N/2)+u)=FTPair(u)-(coef* FTImpair(u));
end
end
end
```

Pour la transformée de Fourier rapide discrète 1D inverse c'est le même principe sauf que le signe du coefficient change :

```
function[image] =TFR1DInv(image1D)
[M,N] = size(image1D);
image=zeros(1, N);
if N==1
   image=image1D;
else

Pair = zeros(1, N / 2);
Impair = zeros(1, N / 2);
for u = 1 : N
 if mod(u, 2) == 0
 Pair(u / 2) = image1D(u);
 else
 Impair( ( u + 1 ) / 2 ) = image1D(u);
 end
end

FTPair=TFR1DInv(Pair);
FTImpair=TFR1DInv(Impair);
for u=1:N/2
    coef = exp( ( 2 * i * pi * u ) / N );
    image(u)=FTPair(u)+(coef * FTImpair(u));
    image(N/2+u)=FTPair(u)-(coef* FTImpair(u));
end
end
end
```

Maintenant lorsqu'on lance nos fonctions sur notre image 1D cela nous donne (on n'oublie pas de diviser par N l'image car c'est le coefficient de passage) :

```
disp(image1D);
image2=TFR1D(image1D);
disp(image2);
image3=TFR1DInv(image2);
[M,N] = size(image3);
image3=image3/N;
disp(image3);
```

```
TFRapide1D
  0     1     3     2     1     0     5     0

1.2929 - 1.1213i   1.0000 - 7.0000i  -2.7071 - 3.1213i  -6.0000 - 0.0000i  -2.7071 + 3.1213i   1.0000 + 7.0000i  -1.2929 + 1.1

0.0000 + 0.0000i   1.0000 + 0.0000i   3.0000 + 0.0000i   2.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   5.0000 + 0.0
```

On voit bien que l'on retombe sur les mêmes valeurs que l'image de base.

Complexité : O(N$\log_2$N) Pour calculer une transformée de Fourier rapide 1D d'une image de taille N nous avons besoin de calculer la transformée de deux tableaux de taille N/2 puis de les rassembler (N calculs). Voici la démonstration de la complexité :

C_N= nombre d'opérations élémentaires pour faire l'algorithme
C_N= 2*$C_{N/2}$+N
2*$C_{N/2}$=4*$C_{N/4}$+N
4*$C_{N/4}$= 8*$C_{N/8}$+N
} n lignes

$C_{N/N/2}$= N$C_{N/N}$+N

C_N= N*C_1+n*N
C_N= N*n*N
or on a posé $N=2^n$
donc $n=\log_2 N$

C_N=N+$\log_2$N
=N(1+$\log_2$N)
$\sim$N*$\log_2$N

On remarque que la complexité est bien inférieure à N^2 la complexité de la transformée de Fourier directe discrète 1D.

IV. Transformée de Fourier rapide discrète 2D et son inverse

La transformée de Fourier rapide discrète 2D c'est en fait une transformée de Fourier rapide discrète 1D appliquée aux lignes et aux colonnes.

On déclare une nouvelle matrice de taille une puissance de 2 (ici 4x4) :

```
image2D=[0 1 5 4  ;  4 2 9 8  ;  5 2 4 6  ;  1 2 5 4 ];
```

Voici donc notre code MATLAB pour la transformée rapide 2D :

```
function[image] =TFR2D(image2D)
[M,N] = size(image2D);
image=zeros(M, N);
ligne = zeros(1, N);
for l = 1 : M
    for c = 1 : N
        ligne(c) = image2D(l, c);
    end
    ligne = TFR1D(ligne);
    for c = 1 : N
    image(l, c) = ligne(c);
    end
end
colonne = zeros(1, M);
for c = 1 : N
    for l = 1 : M
    colonne(l,1) = image2D(l, c);
    end
    colonne = TFR1D(colonne);
    for l = 1 : M
    image(l, c) = colonne(l);
    end
end
end
```

Et son inverse :

```
function[image] =TFR2DInv(image2D)
[M,N] = size(image2D);
image=zeros(M, N);
ligne = zeros(1, N);
for l = 1 : M
    for c = 1 : N
        ligne(1,c) = image2D(l, c);
    end
    ligne = TFR1DInv(ligne);
    for c = 1 : M
    image(l, c) = ligne(1,c);
    end
end
colonne = zeros(1, M);
for c = 1 : N
    for l = 1 : M
    colonne(l,1) = image2D(l, c);
    end
    colonne = TFR1DInv(colonne);
    for l = 1 : M
    image(l, c) = colonne(1,l);
    end
end
end
```

Une fois appliquées à notre image cela nous donne :

```
image2DTF=TFR2D(image2D);
image2DTFI=TFR2DInv(image2DTF);
disp(image2D);
disp(image2DTF);
image3=image2DTFI;
[M,N] = size(image3);
image3=image3/N;
disp(image3);
```

```
    0     1     5     4
    4     2     9     8
    5     2     4     6
    1     2     5     4

 -3.0000 - 5.0000i  -0.0000 - 1.0000i  -4.0000 + 1.0000i  -4.0000 - 2.0000i
  0.0000 - 0.0000i   1.0000 - 0.0000i   5.0000 - 0.0000i   2.0000 - 0.0000i
 -3.0000 + 5.0000i   0.0000 + 1.0000i  -4.0000 - 1.0000i  -4.0000 + 2.0000i
 10.0000 + 0.0000i   7.0000 + 0.0000i  23.0000 + 0.0000i  22.0000 + 0.0000i

  0.0000 + 0.0000i   1.0000 + 0.0000i   5.0000 + 0.0000i   4.0000 + 0.0000i
  4.0000 + 0.0000i   2.0000 + 0.0000i   9.0000 + 0.0000i   8.0000 + 0.0000i
  5.0000 + 0.0000i   2.0000 + 0.0000i   4.0000 + 0.0000i   6.0000 + 0.0000i
  1.0000 - 0.0000i   2.0000 - 0.0000i   5.0000 + 0.0000i   4.0000 + 0.0000i
```

On retombe bien sur les valeurs de notre image de départ.

Complexité : $O(2*N^2\log_2 N)$ On remarque que la complexité est bien inférieure à N^4 la complexité de la transformée de Fourier directe discrète 2D. Le gain de temps de calcul est considérable.