VI Transformée de Fourier discrète et filtrage

- Revisite du filtrage *linéaire* vu précédemment, mais sous l'angle du filtrage « directement » dans le plan de Fourier, i.e. le plan des fréquences spatiales (par le biais de la Transformée de Fourier (TF) bidimensionnelle).

  => de nouveaux horizons vont s'ouvrir à nous en termes d'amélioration/de restoration d'image…

- Illustrations/applications :
  - filtrage passe-bas (afin de « lisser » une image),
  - filtrage passe-haut (afin de faire ressortir les hautes fréquences => par exemple des contours),
  - filtrage passe-bande (pour enlever par ex. des biais périodiques => par exemple un tramage).

- D'autres applications abordées à la fin de ce chapitre (restoration/reconstruction d'image => déconvolution).

(1) TF discrète (TFD, ou DFT en anglais) bidimensionnelle

• Soit $f(x,y)$, avec : *x=0,1,2,…,M-1*, et : *y=0,1,2,…,N-1*, une image digitale de taille *MxN*.

La TFD de $f(x,y)$, $\hat{f}(u,v)$, s'écrit :

$$\hat{f}(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\ \exp\{-\imath 2\pi(ux/M + vy/N)\}$$

Remarque : dans le monde continu, on a :

$$\hat{f}(u,v) = \int \int f(x,y)\ \exp\{-\imath 2\pi(ux + vy)\}\ dx\ dy$$

• *x* et *y* sont les coordonnées spatiales des pixels de l'image $f(x,y)$

• *u* et *v* sont les coordonnées fréquentielles des frequels (frequels = « frequency elements » = les pixels dans le plan de Fourier) de $\hat{f}(u,v)$.

• La TFD inverse s'écrit :

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \hat{f}(u,v)\ \exp\{\imath 2\pi(ux/M + vy/N)\}$$

Même remarque :

$$f(x,y) = \int \int \hat{f}(u,v)\ \exp\{\imath 2\pi(ux + vy)\}\ du\ dv$$

- $\hat{f}(0,0)$ = « composante continu » (vient de l'électronique) = intégrale (somme dans le monde discret) de $f(x,y)$

- Même si $f(x,y)$ est réelle, $\hat{f}(u,v)$ est a priori complexe.

- On peut donc a priori exprimer $\hat{f}(u,v)$ comme :

$$\hat{f}(u,v) = |\hat{f}(u,v)| \exp\{\imath\phi(u,v)\}$$

où $|\hat{f}(u,v)|$ est le module de $\hat{f}(u,v)$ et $\phi(u,v)$ son argument.

$$|\hat{f}(u,v)| = \sqrt{Re^2(\hat{f}(u,v)) + Im^2(\hat{f}(u,v))}$$

$$\phi(u,v) = \arctan \frac{Im(\hat{f}(u,v))}{Re(\hat{f}(u,v))}$$

- Densité spectrale :

$$|\hat{f}(u,v)|^2 = Re^2(\hat{f}(u,v)) + Im^2(\hat{f}(u,v))$$

- Quelques propriétés de la TFD

  - *f* réelle => $Re(\hat{f})$ paire et $Im(\hat{f})$ impaire
  Et : $|\hat{f}(u,v)| = |\hat{f}(-u,-v)|$ module toujours paire
  => Densité spectrale centro-symmétrique

  - *f* paire => $Im(\hat{f}) = 0$

  - La TFD est <u>*circulaire*</u> (i.e. périodique de période *M* en *u* et de période *N* en *v*)

  - La TFD inverse aussi (de période *M* en *x* et *N* en *y)*
  => l'image obtenue par TFD inverse est <u>*périodisée*</u> *!*

  - Ces deux derniers points : car les images et leurs TFD sont échantillonnées.

  Échantillonnage dans le plan direct (respectivement dans le plan de Fourier) <=> périodisation dans le plan de Fourier (respectivement dans le plan direct).

- Calculer la TFD sous Matlab/Octave : en fait calcul d'une FFT (*Fast Fourier Transform*) :

  >> $\hat{f}$ *= fft2(f)*

- Mais la FFT produit un décalage de (*M*/2, *N*/2) => *fftshift* sous Matlab/Octave pour réordonner.

- Quand on va devoir placer l'image initiale dans un tableau plus grand pour le filtrage, on utilisera une syntaxe plus complète pour la FFT :

  >> $\hat{f}$ = *fft2(f, P, Q)*

  où *P* et *Q* plus grands que *M* et *N* (*zero padding*).

- Module de $\hat{f}(u, v)$ obtenu avec la commande *abs* :

  >> *mod = abs($\hat{f}$)*

- Visualiser |FFT| correctement (*frequel* central au centre du tableau visualisé, pas dans les coins) :

  >> *imagesc(abs(fftshift(fft2(f))))*

- Pour mieux voir (dynamique souvent limitée par rapport à la dynamique de la TF) :

  >> *imagesc((abs(fftshift(fft2(f))))^0.5)*

- Opération inverse à *fftshift* : *ifftshift*, ainsi :

  >> $\hat{f}$*=ifftshift(fftshift($\hat{f}$))*

- Pour calculer la phase (ou argument) de $\hat{f}$: *atan2(Im(f),Re(f))*, qui est un tableau d'angles entre -π et π. Autrement dit :

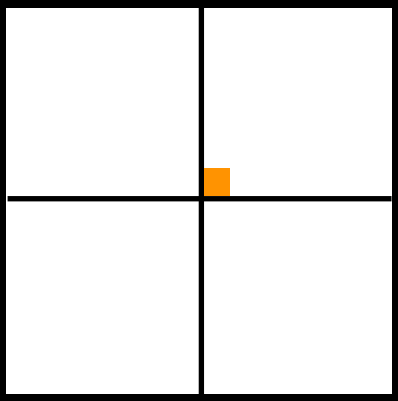
>> phi = atan2(imag($\hat{f}$), real($\hat{f}$))

ou plus directement avec *angle* (ou même *arg*) :

>> phi = angle($\hat{f}$)

(Et on a donc : $\hat{f}$ = *abs($\hat{f}$) x exp(i * angle($\hat{f}$))*)

- Attention, pour la TFD : le centre se situe ici [*M/2+1*, *N/2+1*] si *M* et *N* sont paires :



(si nb impairs (*M* ou *N*) : *floor(M/2)+1*, *floor(N/2)+1*)

- La FFT inverse s'obtient grâce à *ifft2* :

>> f = ifft2($\hat{f}$)

- Attention : *fft2* convertit au passage en classe « double »…

f de type *uint8* —> $\hat{f}$ de type *double* (valeurs réelles en double précision, mais entre 0.0 et 255.0)

=> pour éviter les problèmes : convertir dès le début les images en réels double précision entre 0.0 et 1.0 !

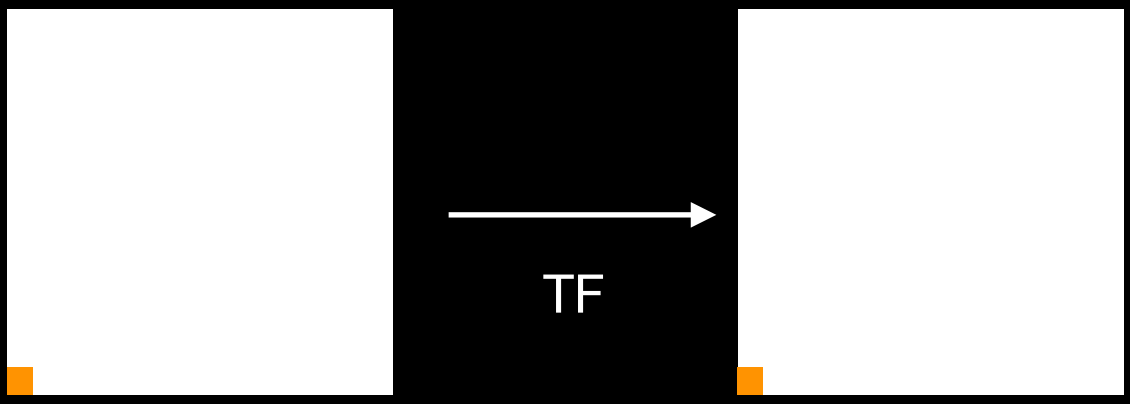
- Unités dans le plan de Fourier :

  - image de longueur $L=N\,\Delta x$, avec N le nb linéaire de pixels et Δx la taille du pixel

  => pas en fréquence $\Delta u = 1/L = 1/(N\,\Delta x)$, en u — et $\Delta v = 1/(N\,\Delta y)$ en v… mais normalement $\Delta x=\Delta y$!

  => vecteur de fréquences spatiales u (ou v) : $u=1/(N\Delta x)\ [-N/2…0…N/2-1]$

  => plus grande (haute!) fréquence spatiale disponible = $1/(N\Delta x) \times N/2 = 1/2\ 1/\Delta x$



$\Delta x=L/N$
$L = N\,\Delta x = N\,L/N$

$\Delta u=1/L$
$N\,\Delta u = N/L = 1/\Delta x$

- Exemple sous Matlab :

  —> FFT bidimensionnelle d'une sinusoïde :

  >> Tx=20; —> période de 20 pixels (tôle ondulée)
  >> x=0:99;
  >> whos x —> vecteur de 100 valeurs de 0 à 99
  *>> I = ones(100,1) * (1 + cos(2*pi*x/Tx));*
  >> imagesc(I), colorbar

  (*>>improfile* sous Matlab, sinon un *plot* pour vérifier la sinusoïde selon l'axe des x)

  >> Ichap = fft2(I);
  >> whos Ichap
  >> figure
  >> subplot(1,2,1), imagesc(I), title('Sinusoïde'), colorbar, axis('square')
  >> subplot(1,2,2), imagesc(abs(fftshift(Ichap))), title('|FFT(Sinusoïde)|'), colorbar, axis('square')

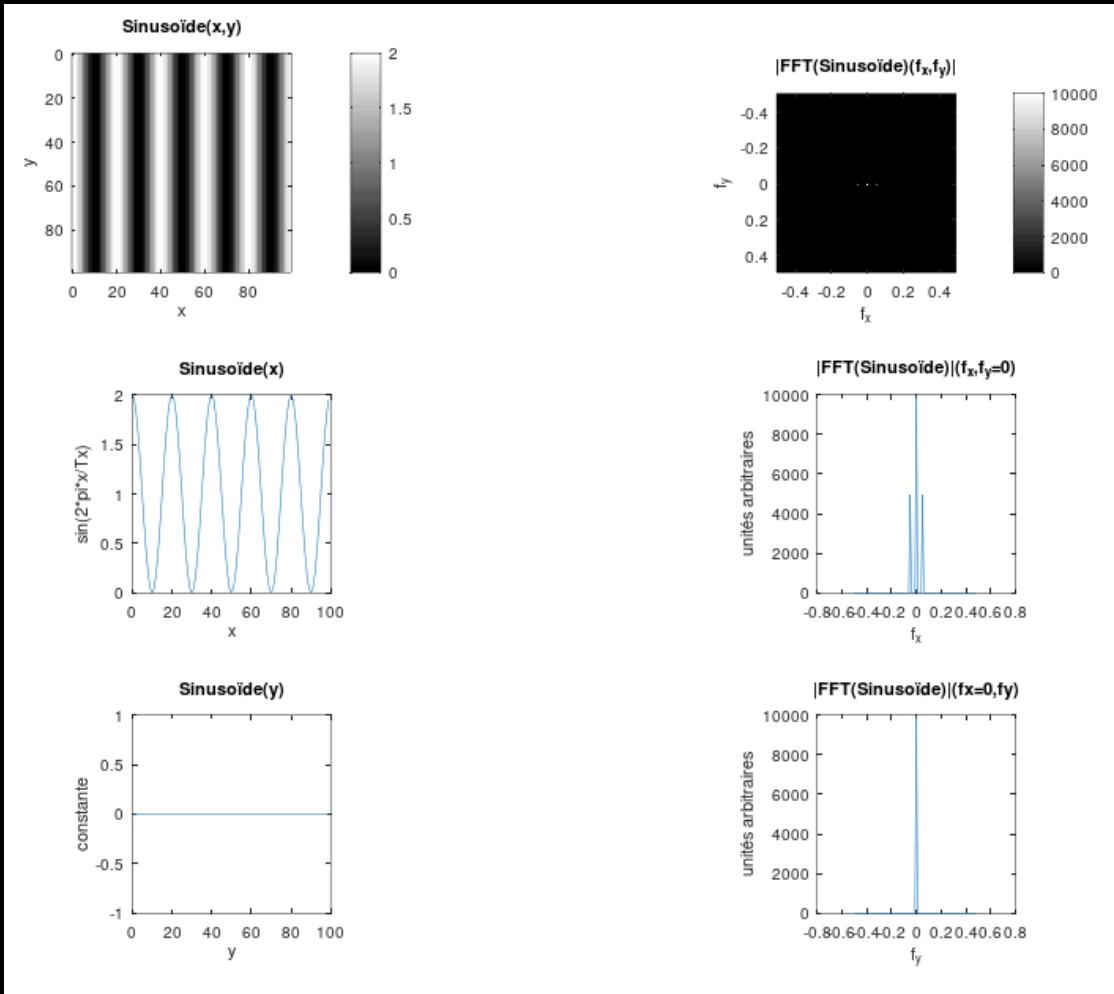
  Remarque : ici, la fonction cosinus étant paire, la partie imaginaire de sa TF est nulle, et donc on aurait pu représenter de manière équivalente la partie réelle.

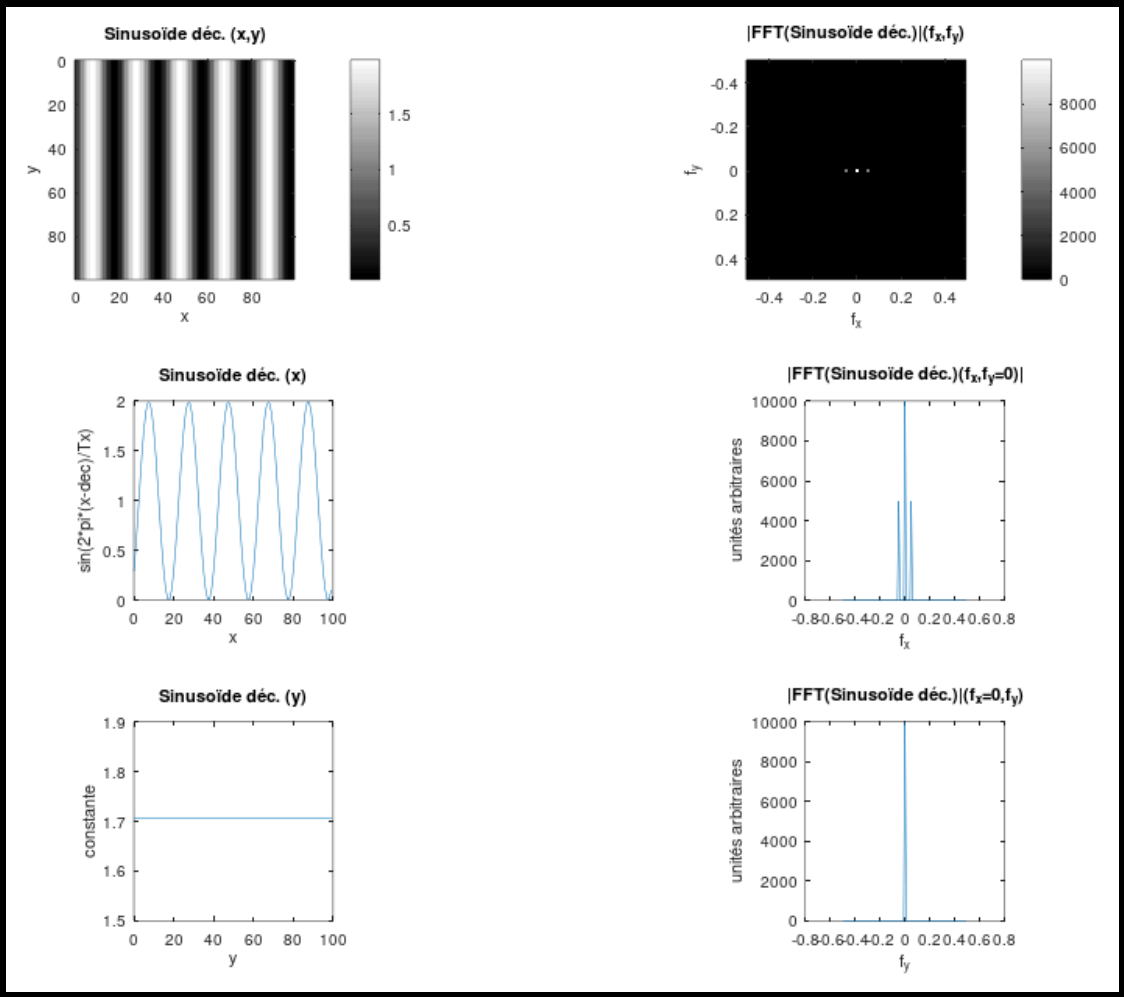
- **Exercice 1** : Reprendre l'exemple ci-dessus. Décaler la sinusoïde de 7.5 pixels. Que se passe-t-il au niveau (du module) de la TF ?

```
clear
close all

% sinusoïde
dim=100;                        % dimension linéaire de l'image
Dx=1;                           % pas (taille du pixel)
x=0:Dx:(dim-1)*Dx;              % vecteur des x
whos x
y=x;                            % vecteur des y

Tx=20;                          % période de la sinusoïde
I=ones(dim,1)*(1+cos(2*pi*x/Tx));
whos I                          % sinusoïde (en fait un cosinus) selon x,
                                % de période Tx et évoluant de 0 à 2

figure(1), colormap('gray')

subplot(3,2,1), imagesc(x,y,I)
title('Sinusoïde(x,y)'), xlabel('x'), ylabel('y'), colorbar, axis('square')

subplot(3,2,3), plot(x,I(dim/2+1,:)), axis('square')
title('Sinusoïde(x)'), xlabel('x'), ylabel('sin(2*pi*x/Tx)')

subplot(3,2,5), plot(y,I(:,dim/2+1)), axis('square')
title('Sinusoïde(y)'), xlabel('y'), ylabel('constante')

```

```
% FFT(sinusoïde)
Ichap=fft2(I);                  % FFT(image)
whos Ichap
Ichapmod=abs(fftshift(Ichap));
whos Ichapmod                   % module de FFT(image)

fx=1/(dim*Dx)*(-dim/2:dim/2-1); fy=fx;
                                % vecteurs des fréquences spatiales

subplot(3,2,2), imagesc(fx,fy,Ichapmod)
title('|FFT(Sinusoïde)(f_x,f_y)|'), xlabel('f_x'), ylabel('f_y')
colorbar, axis('square')

subplot(3,2,4), plot(fx,Ichapmod(dim/2+1,:)), axis('square'),
title('|FFT(Sinusoïde)|(f_x,f_y=0)'), xlabel('f_x'), ylabel('unités arbitraires')

subplot(3,2,6), plot(fy,Ichapmod(:,dim/2+1)), axis('square')
title('|FFT(Sinusoïde)|(fx=0,fy)'), xlabel('f_y'), ylabel('unités arbitraires')

```

```
% cas de la sinusoïde décalée
dec=7.5;
Idec=ones(dim,1)*(1+cos(2*pi*(x-dec)/Tx));
Idecchap=fft2(Idec);
Idecchapmod=abs(fftshift(Idecchap));

figure(2), colormap('gray')

subplot(3,2,1), imagesc(x,y,Idec)
colorbar, axis('square')
title('Sinusoïde déc. (x,y)'), xlabel('x'), ylabel('y')

subplot(3,2,2), imagesc(fx,fy,Idecchapmod)
axis('square'), colorbar
title('|FFT(Sinusoïde déc.)|(f_x,f_y)'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,3), plot(x,Idec(dim/2+1,:))
axis('square')
title('Sinusoïde déc. (x)'), xlabel('x'), ylabel('sin(2*pi*(x-dec)/Tx)')

subplot(3,2,4), plot(fx,Idecchapmod(dim/2+1,:))
axis('square')
title('|FFT(Sinusoïde déc.)(f_x,f_y=0)|')
xlabel('f_x'), ylabel('unités arbitraires')

subplot(3,2,5), plot(y,Idec(:,dim/2+1))
axis('square')
title('Sinusoïde déc. (y)'), xlabel('y'), ylabel('constante')

subplot(3,2,6), plot(fy,Ichapmod(:,dim/2+1))
axis('square')
title('|FFT(Sinusoïde déc.)|(f_x=0,f_y)')
xlabel('f_y'), ylabel('unités arbitraires')
```




Sinusoïde déc. (x,y)
y
x
|FFT(Sinusoïde déc.)|(fx,fy)
fy
fx
Sinusoïde déc. (x)
sin(2*pi*(x-déc)/Tx)
x
|FFT(Sinusoïde déc.)(fx,fy=0)|
unités arbitraires
fx
Sinusoïde déc. (y)
constante
y
|FFT(Sinusoïde déc.)|(fx=0,fy)
unités arbitraires
fy

Autres exemples communs/utiles

—> Porte bidimensionnelle

$$f(x,y) = \Pi\left(\frac{x}{a}\right)\ \Pi\left(\frac{y}{b}\right)$$

TF(porte en x) = sinc(u a) ; TF(porte en y) = sinc(v b)

$$\hat{f}(u,v) = \mathrm{sinc}(u\,a)\ \mathrm{sinc}(v\,b)$$

Remarque 1 : ∏(x/a) et ∏(y/b) sont des fonctions séparables en x et en y => pas de convolution dans le plan de Fourier ici !

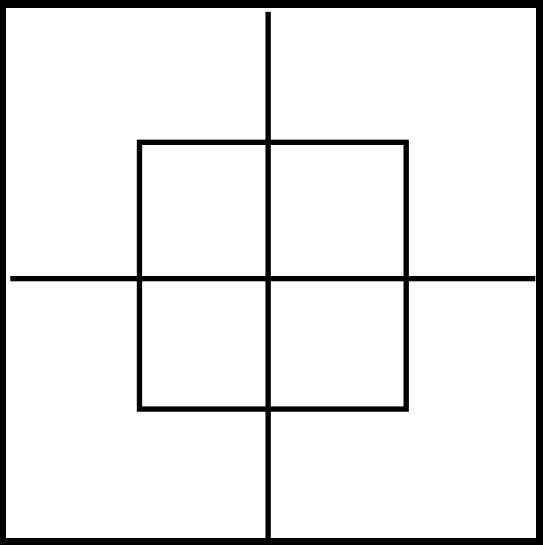
Remarque 2 : Les portes en x et y peuvent décrire le fait qu'une image est de taille *a x b* (en pixels).

Remarque 3 : ∏(x/a) = porte de largeur *a* en *x*.
sinc(u a) de largeur inversement prop. à *a*.

Quand *a* augmente, le pic du *sinc* augmente, mais sa largeur diminue => plus la porte est large, plus le *sinc* dans Fourier est étroit.

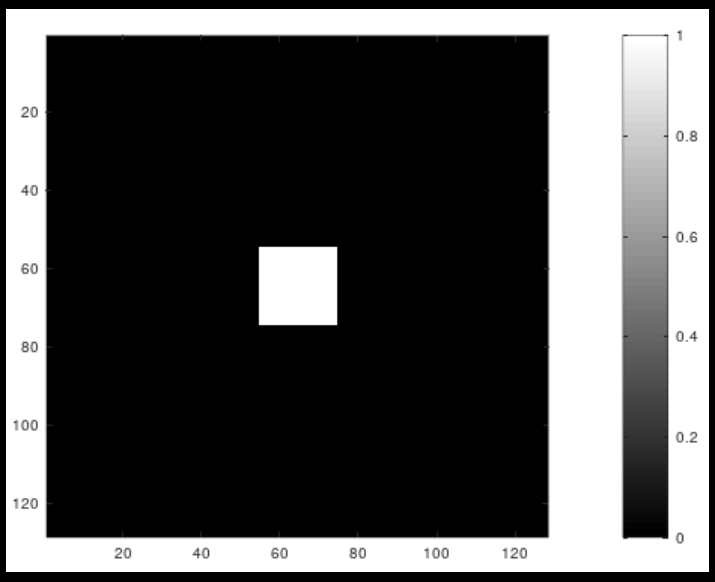
sous Matlab/Octave :

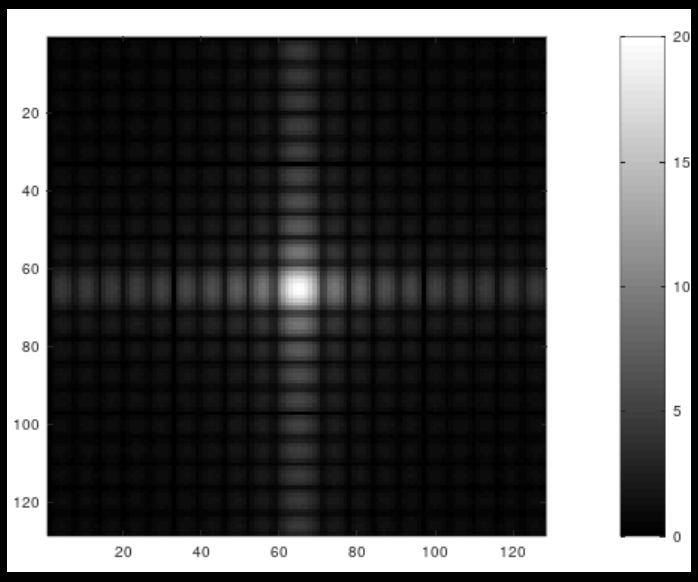
```
>> dim=128; a=20; b=20;
>> P=zeros(dim,dim);
```



1 dim/2 dim/2+1 dim

```
>>  P(dim/2-a/2+1:dim/2+a/2,dim/2-b/2+1:dim/2+b/2)=1;
>> colormap(gray)
>> imagesc(P), axis('square')
>> Pchap=fft2(P);
>> imagesc((abs(fftshift(Pchap))).^0.5), colorbar
```

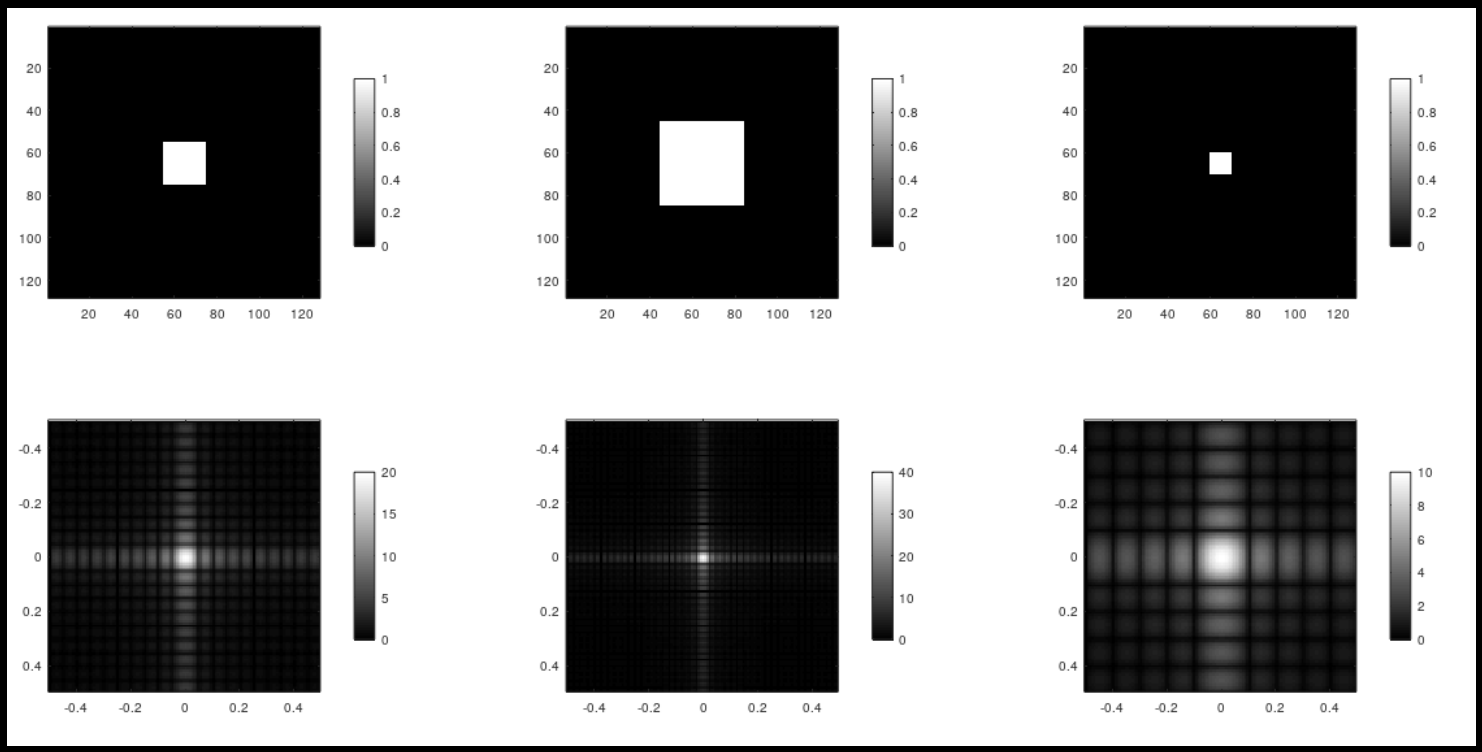


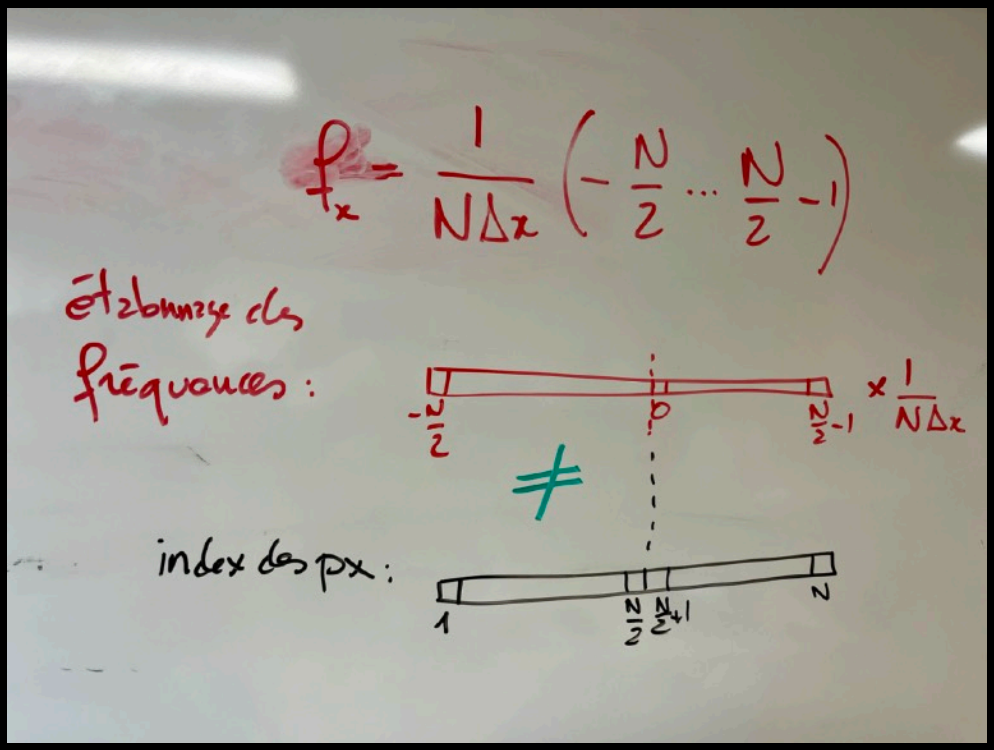


Exercice 2 : Reprendre l'exemple précédent et augmenter/diminuer *a* et *b*. Que remarque-t-on ?
(Prendre par exemple 3 valeurs : 20, 10, 40)
(Mettre aussi les bonnes échelles de fréquences spatiales dans Fourier…)

```
clear
close all

dim=128; fx=1/(dim*1)*(-dim/2:dim/2-1); fy=fx;

% 1er exemple
P20=zeros(dim,dim); a=20; b=20;
P20(dim/2-a/2+1:dim/2+a/2, dim/2-b/2+1:dim/2+b/2)=1.0;
Pchapmod20=abs(fftshift(fft2(P20)));

% 2me exemple
P40=zeros(dim,dim); a=40; b=40;
P40(dim/2-a/2+1:dim/2+a/2, dim/2-b/2+1:dim/2+b/2)=1.0;
Pchapmod40=abs(fftshift(fft2(P40)));

% 3me exemple
P10=zeros(dim,dim); a=10; b=10;
P10(dim/2-a/2+1:dim/2+a/2, dim/2-b/2+1:dim/2+b/2)=1.0;
Pchapmod10=abs(fftshift(fft2(P10)));

```

```
% figure finale
figure, colormap('gray')

subplot(2,3,1), imagesc(P20)
colorbar, axis('square')
title('Porte(x/20,y/20)'), xlabel('x'), ylabel('y')

subplot(2,3,2), imagesc(P40)
colorbar, axis('square')
title('Porte(x/40,y/40)'), xlabel('x'), ylabel('y')

subplot(2,3,3), imagesc(P10)
colorbar, axis('square')
title('Porte(x/10,y/10)'), xlabel('x'), ylabel('y')

subplot(2,3,4), imagesc(fx,fy,Pchapmod20.^.5)
colorbar, axis('square')
title('sqrt(|FFT(Porte_{20})|(f_x,f_y))'), xlabel('f_x'), ylabel('f_y')

subplot(2,3,5), imagesc(fx,fy,Pchapmod40.^.5)
colorbar, axis('square')
title('sqrt(|FFT(Porte_{40})|(f_x,f_y))'), xlabel('f_x'), ylabel('f_y')

subplot(2,3,6), imagesc(fx,fy,Pchapmod10.^.5)
colorbar, axis('square')
title('sqrt(|FFT(Porte_{10})|(f_x,f_y))'), xlabel('f_x'), ylabel('f_y')
```


20 40 60 80 100 120
1 0.8 0.6 0.4 0.2 0
-0.4 -0.2 0 0.2 0.4
20 15 10 5 0
40 30 20 10 0
10 8 6 4 2 0


$f_x = \frac{1}{N\Delta x}\left(-\frac{N}{2} \dots \frac{N}{2}-1\right)$
étalonnage des fréquences :
$-\frac{N}{2}$
0
$\frac{N}{2}-1$
$\times \frac{1}{N\Delta x}$
≠
index des px :
1
$\frac{N}{2}$
$\frac{N}{2}+1$
N

—> Gaussienne bidimensionnelle

$$f(x,y) = \exp\{-\pi \frac{x^2}{a^2}\} \ \exp\{-\pi \frac{y^2}{b^2}\}$$

$$\hat{f}(u,v) = a \ \exp\{-\pi(a\,u)^2\} \ b \ \exp\{-\pi(b\,v)^2\}$$

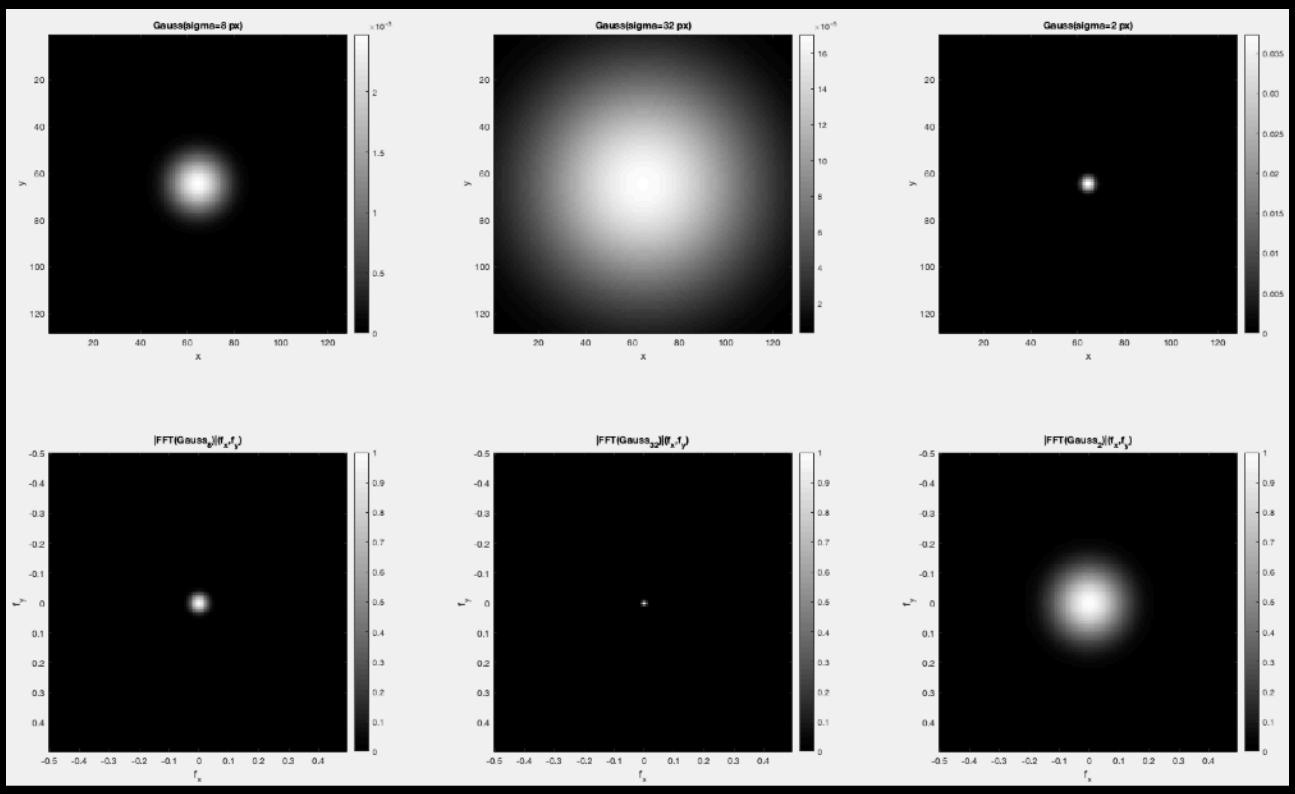
sous Matlab/Octave :

>> dim=128; sigma=10;
>> G = fspecial('gaussian', dim, sigma);
>> …

Exercice 3 : Comme l'exercice 2 mais avec une Gaussienne…
(En prenant trois valeurs de sigma et toujours en étalonnant correctement les fréquences spatiales.)

```
clear
close all
pkg load image

dim=128; fx=1/(dim*1)*(-dim/2:dim/2-1); fy=fx;

% 1er exemple
a=8;
G8=fspecial('gaussian',dim,a);
Gchapmod8=abs(fftshift(fft2(G8)));

% 2me exemple
a=32;
G32=fspecial('gaussian',dim,a);
Gchapmod32=abs(fftshift(fft2(G32)));

% 3me exemple
a=2;
G2=fspecial('gaussian',dim,a);
Gchapmod2=abs(fftshift(fft2(G2)));

```

```
% figure finale
figure, colormap('gray')

subplot(2,3,1), imagesc(G8)
colorbar, axis('square')
title('Gauss(sigma=8 px)'), xlabel('x'), ylabel('y')

subplot(2,3,2), imagesc(G32)
colorbar, axis('square')
title('Gauss(sigma=32 px)'), xlabel('x'), ylabel('y')

subplot(2,3,3), imagesc(G2)
colorbar, axis('square')
title('Gauss(sigma=2 px)'), xlabel('x'), ylabel('y')

subplot(2,3,4), imagesc(fx,fy,Gchapmod8)
colorbar, axis('square')
title('|FFT(Gauss_{8})|(f_x,f_y)'), xlabel('f_x'), ylabel('f_y')

subplot(2,3,5), imagesc(fx,fy,Gchapmod32)
colorbar, axis('square')
title('|FFT(Gauss_{32})|(f_x,f_y)'), xlabel('f_x'), ylabel('f_y')

subplot(2,3,6), imagesc(fx,fy,Gchapmod2)
colorbar, axis('square')
title('|FFT(Gauss_{2})|(f_x,f_y)'), xlabel('f_x'), ylabel('f_y')
```



—> Dirac :

$\partial(x,y) \xrightarrow{TF} \mathbf{1}(u,v)$

—> Continu :

$a\,\mathbf{1}(x,y) \xrightarrow{TF} a\,\partial(u,v)$

—> Peigne de Dirac :

$\text{Ш}_{(a)}(x) \,.\, \text{Ш}_{(b)}(y) \xrightarrow{TF} \text{Ш}_{(1/a)}(u) \,.\, \text{Ш}_{(1/b)}(v)$

La fonction *Sha* de période *a* en *x* ($\text{Ш}_{(a)}(x)$) et de période *b* en *y* ($\text{Ш}_{(b)}(y)$) décrit, notamment, l'échantillonnage d'une image (les pixels!)…

(on prend dans la suite $\Delta y = \Delta x$ = taille du pixel)

=> image discrète (x,y) = image continue (x,y) $\bullet$ $(\text{Ш}_{(\Delta x)}(x) \,.\, \text{Ш}_{(\Delta x)}(y))$

=> *DFT*(img discrète) (u,v) = *DFT*(img continue) (u,v) $*$ $(\text{Ш}_{(1/\Delta x)}(u) \,.\, \text{Ш}_{(1/\Delta x)}(v))$

(où $\bullet$ décrit le produit et $*$ le produit de convolution)

Or : $1/\Delta x$ = largeur de toute la *DFT*

Donc, on a : échantillonnage dans le plan direct => périodisation dans le plan de Fourier !

- **Trois propriétés remarquables de la TF**

—> Dilatation

$$f\left(\frac{x}{a}, \frac{y}{b}\right) - \text{par } TF \rightarrow |a|\ |b|\ \hat{f}(a\,u, b\,v)$$

Ce qui est étroit dans l'espace direct est large dans l'espace de Fourier, et vice versa.

—> Translation

$$f(x + x_0, y + y_0) - \text{par } TF \rightarrow \hat{f}(u, v)\ \exp\{-2\imath\pi(x_0\,u + y_0\,v)\}$$

car on a au passage…

$$f(x + x_0, y + y_0) = f(x, y) * \delta((x + x_0, y + y_0)$$

Remarque 1 : ceci constitue un excellent moyen de ***décaler*** ou ***interpoler*** une image… (simple multiplication de la TF par un terme de phase.)
Remarque 2 : le module est inchangé.

—> Convolution

$$f(x, y).g(x, y) - \overset{\text{TF}}{\rightarrow} \hat{f}(u, v) * \hat{g}(u, v)$$

$$f(x, y) * g(x, y) - \overset{\text{TF}}{\rightarrow} \hat{f}(u, v).\hat{g}(u, v)$$

C'est la propriété que l'on va utiliser pour le filtrage (et qui est au cœur de la déconvolution).

Exercice 4 : Décaler la Gaussienne bidimensionnelle de l'exercice précédent (avec *sigma*=10 px) de 10,4 px en *x* et -10,4 px en *y*, par TF.

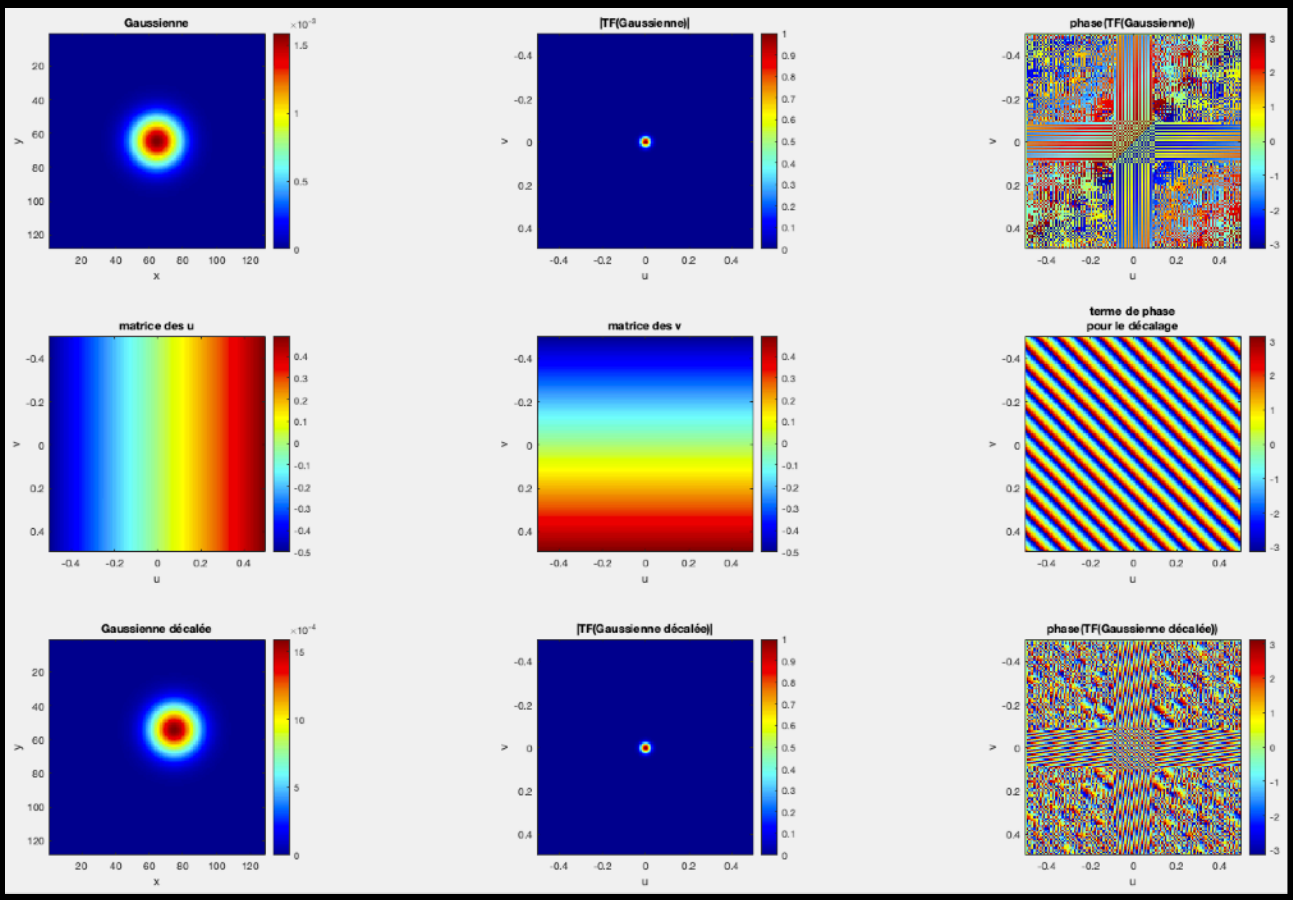
Étapes :
(1) création de la Gaussienne (plan direct)
(2) calculer le terme de phase (plan de Fourier)
(3) appliquer le terme de phase (plan de Fourier)
(4) revenir dans le plan direct

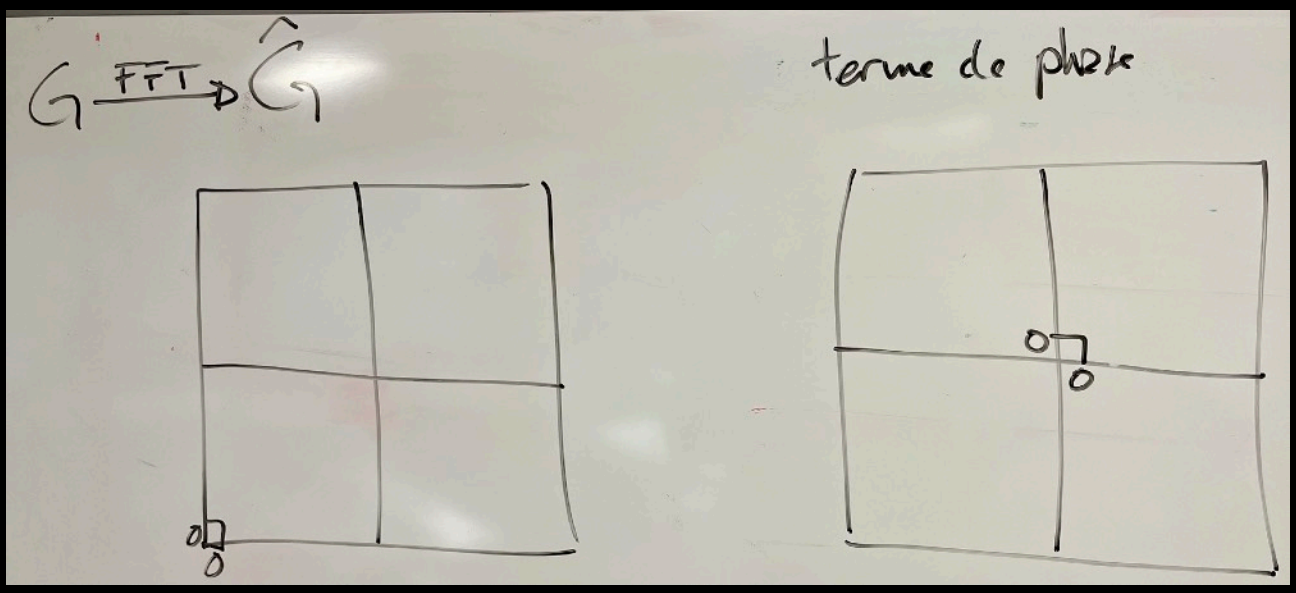
L'étape (2) revient à écrire correctement le terme de phase *exp(-2 i π (xo u + yo v)* dans Fourier… terme qui doit être un tableau de mêmes dimensions que l'image de départ (et donc sa TF et donc la phase de cette TF). Ici *xo* et *yo* sont les décalages respectivement en *x* et en *y*, et ce sont des nombres, pas des tableaux. Par contre, *u* et *v* doivent décrire toutes les valeurs des fréquences dans le plan de Fourier… Ce sont donc des tableaux de mêmes dimensions que l'image de départ. Par exemple : *u=ones(dim, 1)*fx*, où : *fx=1/(dim*1)*(-dim/2:dim/2-1)*.

Attention : on est ici dans le plan de la **FFT**, avec son décalage d'un demi-tableau à droite et vers le haut, il faut donc utiliser *fftshift* de manière à avoir les fréquences des tableaux de *u* et de *v* en face de celles de la FFT de l'image.

```
% préliminaires
clear
close all
%pkg load image

dim=128;                                % taille des images
fx=1/(dim*1)*(-dim/2:dim/2-1); fy=fx; % vecteur des fréquences :
                                        % 1/(N Δx)*(-N/2 .. 0 .. N/2-1)
                                        % on peut aussi écrire :
                                        % fx=((0:dim-1)-dim/2)/dim; fy=fx;
% Gaussienne
a=10;
G=fspecial('gaussian',dim,a);           % Gaussienne
Gchap = fft2(G);                        % TF(Gaussienne)

% terme de phase
decx= 10.4;                             % décalage en x
decy=-10.4;                             % décalage en y
uo=ones(dim,1)*fx;                      % matrice des u (par colonne)
vo=uo';                                 % matrice des v (par ligne)
u=fftshift(uo);                         % car le plan de la FFT est "shifté"
v=fftshift(vo);                         % d'un demi-tableau en u et en v
decfou=exp(-complex(0,1)*2*pi*(u*decx+v*decy));
                                        % TF(décalage) ; complex(0,1)=i
% décalage
Gdecchap = Gchap.*decfou;               % TF(Gaussienne décalée)
Gdec=real(ifft2(Gdecchap));             % retour dans le plan direct
```

```
% figure finale
figure(1), colormap('jet')              % colormap "jet" (voir help colormap)

subplot(3,3,1), imagesc(G)
colorbar, axis('square')
title('Gaussienne'), xlabel('x'), ylabel('y')

subplot(3,3,2), imagesc(fx, fy, abs(fftshift(Gchap)))
colorbar, axis('square')
title('|TF(Gaussienne)|'), xlabel('u'), ylabel('v')

subplot(3,3,3), imagesc(fx, fy, angle(fftshift(Gchap)))
colorbar, axis('square')
title('phase(TF(Gaussienne))'), xlabel('u'), ylabel('v')

subplot(3,3,4), imagesc(fx, fy, uo)
colorbar, axis('square')
title('matrice des u'), xlabel('u'), ylabel('v')

subplot(3,3,5), imagesc(fx, fy, vo)
colorbar, axis('square')
title('matrice des v'), xlabel('u'), ylabel('v')

subplot(3,3,6), imagesc(fx, fy, angle(fftshift(decfou)))
colorbar, axis('square')
title({'terme de phase'; 'pour le décalage'}), xlabel('u'), ylabel('v')

subplot(3,3,7), imagesc(Gdec)
colorbar, axis('square')
title('Gaussienne décalée'), xlabel('x'), ylabel('y')

subplot(3,3,8), imagesc(fx, fy, abs(fftshift(Gdecchap)))
colorbar, axis('square')
title('|TF(Gaussienne décalée)|'), xlabel('u'), ylabel('v')

subplot(3,3,9), imagesc(fx, fy, angle(fftshift(Gdecchap)))
colorbar, axis('square')
title('phase(TF(Gaussienne décalée))'), xlabel('u'), ylabel('v')
```


Gaussienne
|TF(Gaussienne)|
phase(TF(Gaussienne))
matrice des u
matrice des v
terme de phase
pour le décalage
Gaussienne décalée
|TF(Gaussienne décalée)|
phase(TF(Gaussienne décalée))
x
y
u
v


FFT
terme de phase

(2) Filtrage dans le plan (complexe) de Fourier

- On utilise le théorème de convolution de la TF :

$f * h \xrightarrow{TF} \hat{f} \bullet \hat{h}$
$f \bullet h \xrightarrow{TF} \hat{f} * \hat{h}$

avec *f* = image, *h* = filtre => $\hat{h}$ = fonction de transfert de *h*

h = *filtre* (et $\hat{h}$ sert de masque dans le plan de Fourier)

L'idée ici est de modeler $\hat{h}$ *(à travers son module)* afin de modifier la distribution des fréquences spatiales présentes dans l'image *f*.

- Filtrage passe-bas

  On fait diminuer les valeurs de $\hat{h}$ au fur et à mesure que l'on s'éloigne du centre *(0,0)* des fréquences spatiales, favorisant ainsi les plus basses fréquences en atténuant les plus hautes.

  Cette distribution peut-être brutale (on « coupe » alors les fréquences plus hautes qu'une certaine fréquence **de coupure** à l'aide d'une porte de largeur $(2\ u_c, 2\ v_c)$ autour du frequel *(0,0)*, où : u_c est la fréquence de coupure en *u*, v_c celle en *v.*

Applications : supprimer les détails fins, réduire le bruit.

Remarque : on prend en général des fonctions de transfert $\hat{h}$ autour du frequel *(0,0)*.

Exemple : revisite du filtrage Gaussien (où l'on cherche à faire le filtrage Gaussien du chapitre 3 mais en passant par Fourier…)

```
>> dim=128;
>> I=zeros(dim,dim);
>> I(:,1:dim/2) = 0.6;
>> figure, colormap(gray)
>> subplot(3,2,1), imagesc(I), colorbar, title('img')
```

—

```
>> Ichap = fft2(I);
>> whos Ichap;
>> subplot(3,2,2), imagesc(abs(fftshift(Ichap)).^0.5),
colorbar, title('|FFT(image)|')
```

—

```
>> h = fspecial('gaussian', dim, 0.849);
>> subplot(3,2,3), imagesc(h.^0.5), colorbar,
title('Gauss')
```

—

```
>> hchap = fft2(h);
>> whos hchap;
>> subplot(3,2,4),imagesc(abs(fftshift(hchap)).^0.5),
colorbar, title('|FFT(Gauss)|')
```

—

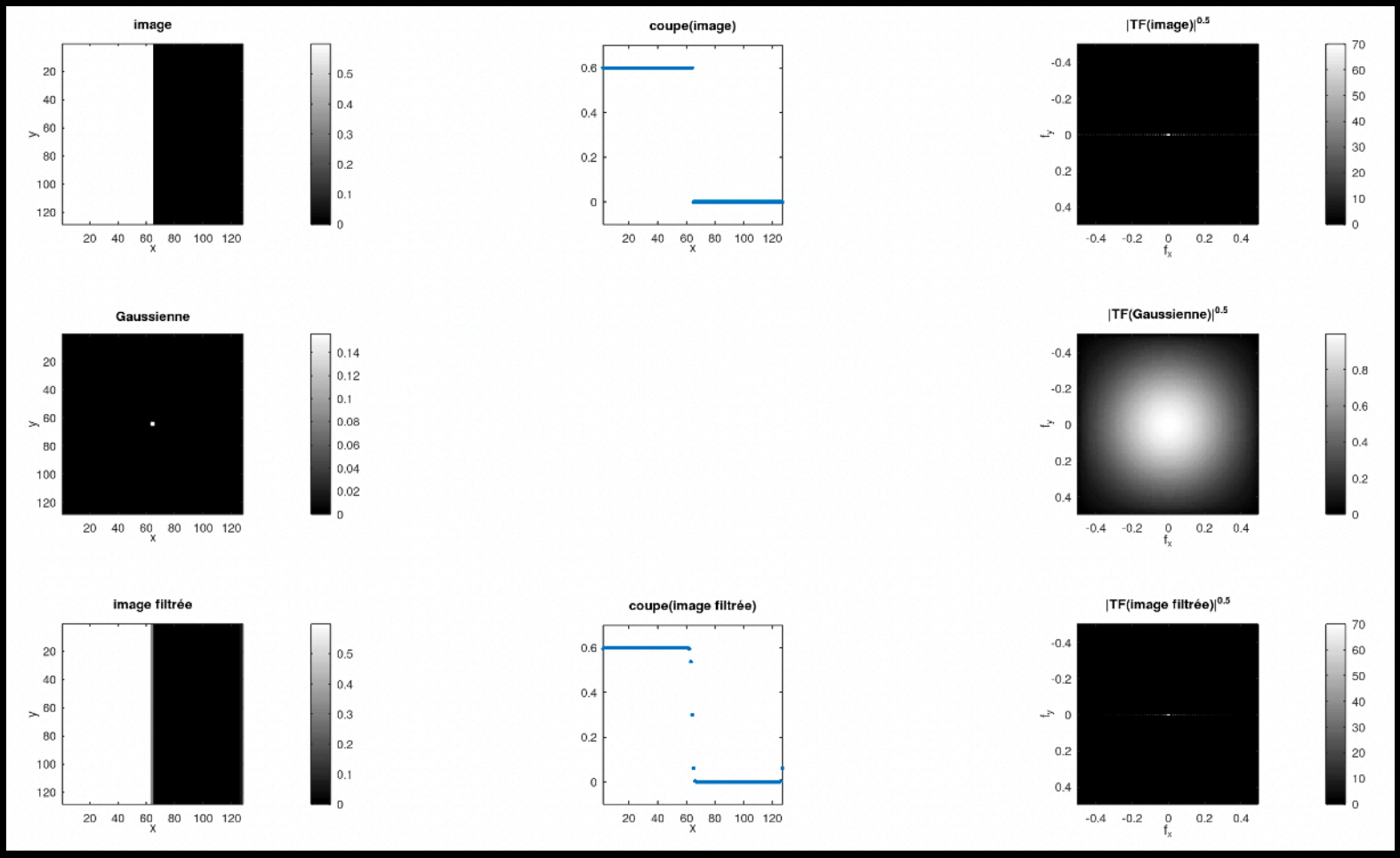
*>> Ichapfilt = hchap.*Ichap;*
>> whos Ichapfilt
>> subplot(3,2,6),
imagesc(abs(fftshift(Ichapfilt)).^0.5), colorbar, title('|
FFT(image filtrée)|')

—

>> Ifilt = ifftshift(ifft2(Ichapfilt));
>> whos(Ifilt)
>> subplot(3,2,5), imagesc(real(Ifilt)), colorbar,
title('image filtrée')

```
clear
close all
pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('gray')

subplot(3,3,1), imagesc(I)
colorbar, axis('square')
title('image'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(64,:),'.')
axis('square')
title('coupe(image)'), xlim([2,dim-1]), ylim([-.1 .7]), xlabel('x')

% FFT(image)
Ichap=fft2(I); Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5)
colorbar, axis('square')
title('|TF(image)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% filtre
h=fspecial('gaussian',dim,.849);
subplot(3,3,4), imagesc(h)
colorbar, axis('square')
title('Gaussienne'),  xlabel('x'), ylabel('y')

% => fct de transfert
hchap=fft2(h); hchapmod=abs(fftshift(hchap));
subplot(3,3,6), imagesc(fx,fy,hchapmod.^.5)
colorbar, axis('square')
title('|TF(Gaussienne)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

```

```
% => fct de transfert
hchap=fft2(h); hchapmod=abs(fftshift(hchap));
subplot(3,3,6), imagesc(fx,fy,hchapmod.^.5)
colorbar, axis('square')
title('|TF(Gaussienne)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=hchap.*Ichap; Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5)
colorbar, axis('square')
title('|TF(image filtrée)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour vers l'espace réel
Ifilt=ifftshift(ifft2(Ichapfilt));
subplot(3,3,7), imagesc(real(Ifilt))
axis('square'), colorbar
title('image filtrée'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(real(Ifilt(64,:)),'.')
axis('square'), xlim([2,dim-1]), ylim([-.1 .7])
title('coupe(image filtrée)'), xlabel('x')
```



Exercice 5bis : Reprendre l'exemple ci-avant et diminuer la largeur de $\hat{h}$ d'un facteur 10. Que se passe-t-il ?

```
clear
close all
pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('cubehelix')

subplot(3,3,1), imagesc(I)
colorbar, axis('square')
title('image'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(64,:),'.')
axis('square')
title('coupe(image)'), xlim([2,dim-1]), ylim([-.1 .7]), xlabel('x')

% FFT(image)
Ichap=fft2(I); Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5)
colorbar, axis('square')
title('|TF(image)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% filtre
h=fspecial('gaussian',dim,.849*10);
subplot(3,3,4), imagesc(h)
colorbar, axis('square')
title('Gaussienne'),  xlabel('x'), ylabel('y')

% => fct de transfert
hchap=fft2(h); hchapmod=abs(fftshift(hchap));
subplot(3,3,6), imagesc(fx,fy,hchapmod.^.5)
colorbar, axis('square')
title('|TF(Gaussienne)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=hchap.*Ichap; Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5)
colorbar, axis('square')
title('|TF(image filtrée)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour vers l'espace réel
Ifilt=ifftshift(ifft2(Ichapfilt));
subplot(3,3,7), imagesc(real(Ifilt))
axis('square'), colorbar
title('coupe(image filtrée)'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(real(Ifilt(64,:)),'.')
axis('square'), xlim([2,dim-1]), ylim([-.1 .7])
title('coupe(image filtrée)'), xlabel('x')
```

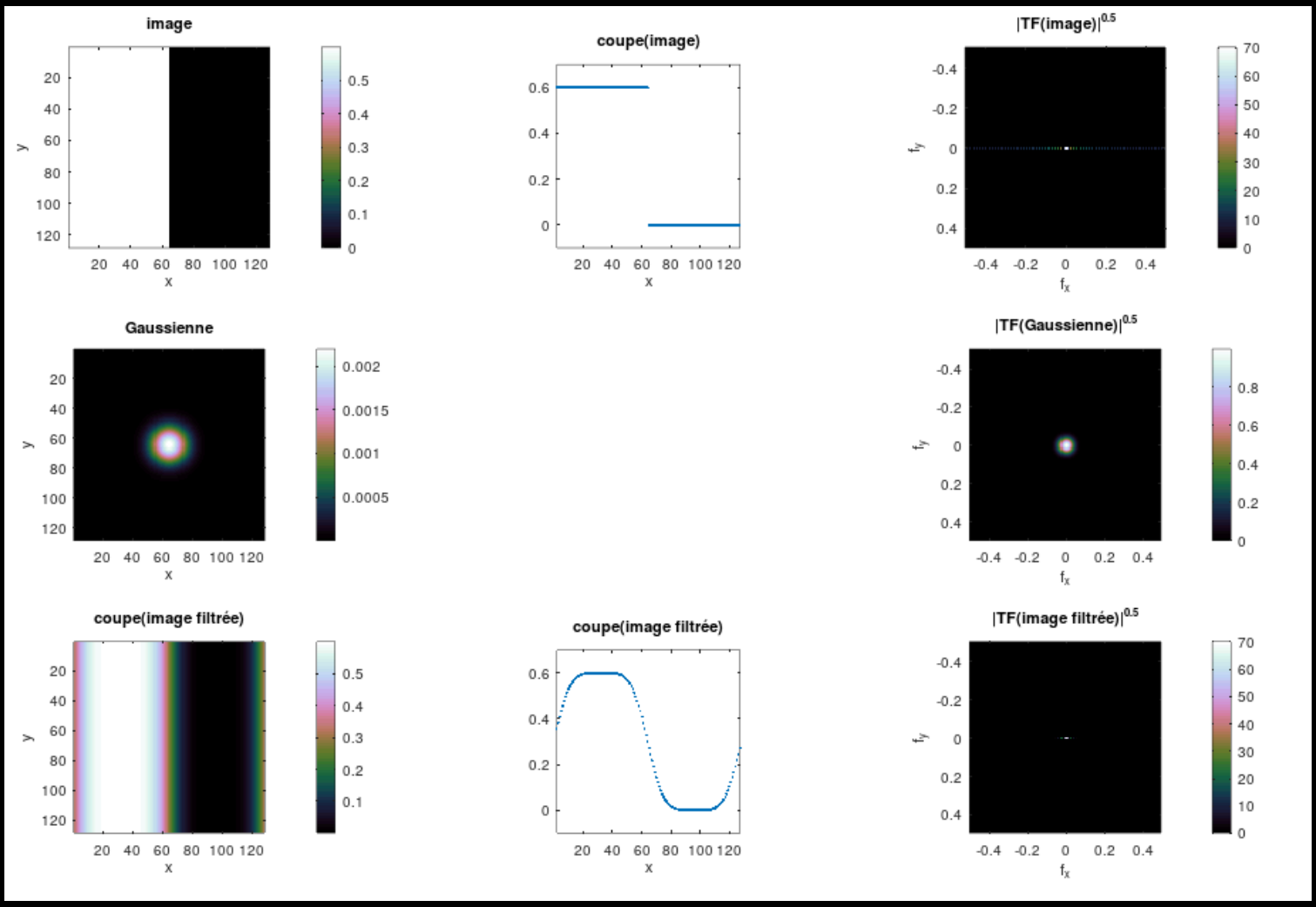


image
coupe(image)
|TF(image)|^0.5
Gaussienne
|TF(Gaussienne)|^0.5
coupe(image filtrée)
coupe(image filtrée)
|TF(image filtrée)|^0.5
x
y
f_x
f_y

Exercice 6 : Faire de même avec la moyenne glissante, en 3x3 puis 5x5.

```
clear
close all
pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;  % image = marche 0.6/0
fx=((0:dim-1)-dim/2)/dim; fy=fx;              % étalonnage des fréquences

figure, colormap('bone')                      % colormap "bone" (vor help)

subplot(3,3,1), imagesc(I)
colorbar, axis('square')
title('image'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(dim/2,:),'.')
title('coupe(image)'), xlabel('x')
axis('square'), xlim([2,dim-2]), ylim([-.1 .7])

% FFT(image)
Ichap=fft2(I);                                % FFT(image)
Ichapmod=abs(fftshift(Ichap));                % module réordonné
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5)
colorbar, axis('square')
title('|TF(img)|'), xlabel('f_x'), ylabel('f_y')

% filtre
nn=5; h=fspecial('average',nn);               % moyenne glissante 3x3 ou 5x5
% on peut utiliser aussi ones(nn,nn).
% et on peut ensuite recentrer le filtre h dans un tableau dim*dim "à la main,
% ou utiliser la commande "padarray", mais ce n'est pas nécessaire puisque l'on
% pourra tout aussi bien faire ça au vol avec la commande FFT2.
```

```
% => fct de transfert
hchap=fft2(h,dim,dim);                        % masque=FFT(filtre)
hchapmod=abs(fftshift(hchap));                % module
subplot(3,3,6), imagesc(fx,fy,hchapmod)
colorbar, axis('square')
title('|TF(Moy.)|'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=Ichap.*hchap;                       % filtrage
Ichapfiltmod=abs(fftshift(Ichapfilt));        % module du résultat dans Fourier
subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5)
colorbar, axis('square')
title('|TF(img filt.)|'), xlabel('f_x'), ylabel('f_y')

% retour vers l'espace réel
Ifilt=real(ifft2(Ichapfilt));                 % image filtrée
subplot(3,3,7), imagesc(Ifilt)
colorbar, axis('square')
title('coupe(img filt.)'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(Ifilt(64,:),'.')
title('coupe(img filt.)'), xlabel('x')
axis('square'), xlim([2,dim-2]), ylim([-.1 .7])
```

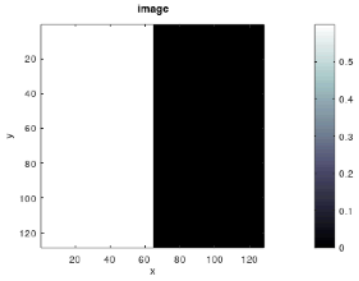
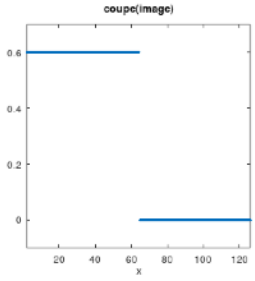
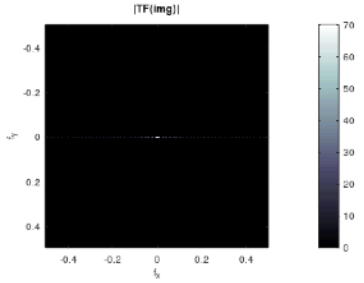
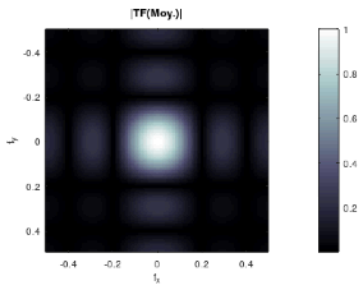
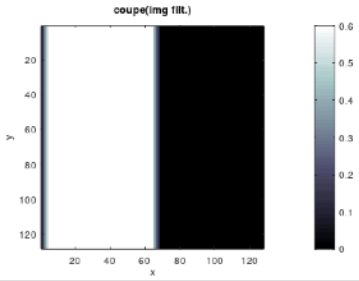


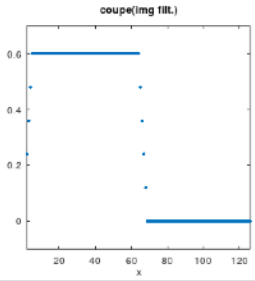
image
y
x

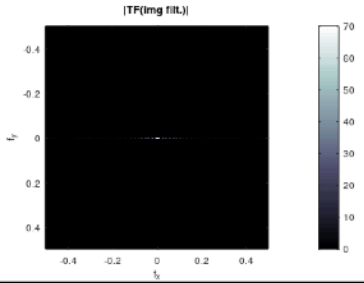

coupe(image)
x


|TF(img)|
f_y
f_x


|TF(Moy.)|
f_y
f_x


coupe(img filt.)
y
x

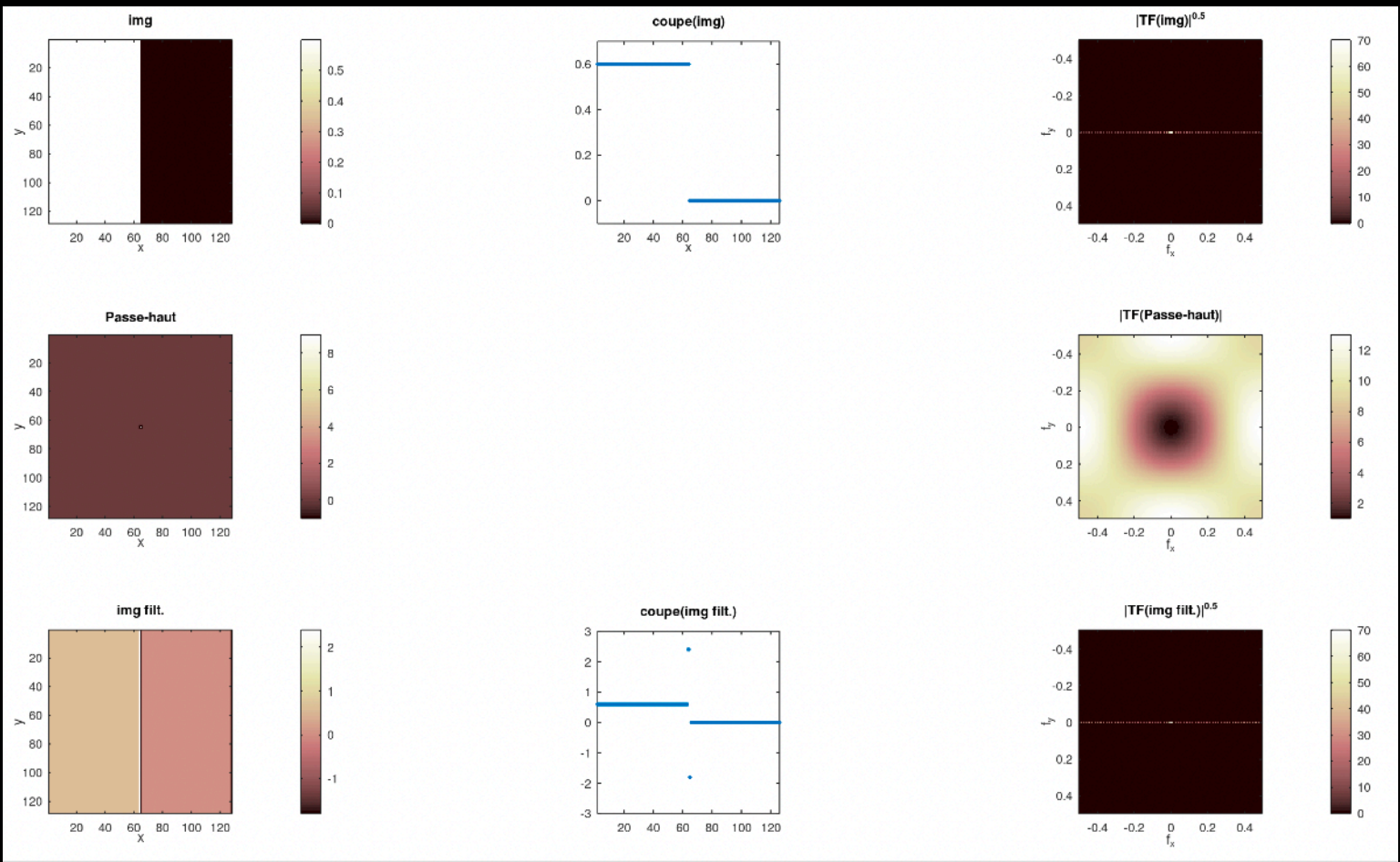

coupe(img filt.)
x


|TF(img filt.)|
f_y
f_x

Exercice 7 : Faire de même avec le filtre passe-haut vu au chapitre 3.

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

```
clear
close all
pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;      % image=marche 0.6/0
fx=((0:dim-1)-dim/2)/dim; fy=fx;                  % étalonnage des fréquences

figure, colormap('pink')

subplot(3,3,1), imagesc(I), colorbar, axis('square')
title('img'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(64,:),'.'), axis('square')
title('coupe(img)'),  xlim([2,dim-2]), ylim([-.1 .7]), xlabel('x')

% FFT(image)
Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));

subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5), colorbar, axis('square')
title('|TF(img)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% filtre => fct de transfert
h=-ones(3,3); h(2,2)=9;                           % filtre passe-haut
hchap=fft2(h,dim,dim);                            % fct de transfert corr.
hchapmod=abs(fftshift(hchap));
subplot(3,3,6), imagesc(fx,fy,hchapmod), colorbar, axis('square')
title('|TF(Passe-haut)|'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=Ichap.*hchap;
Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5), colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% image filtrée
Ifilt=real(ifft2(Ichapfilt));
subplot(3,3,7), imagesc(Ifilt), colorbar, axis('square')
title('img filt.'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(Ifilt(64,:),'.'), title('coupe(img filt.)')
axis('square'), xlim([2,dim-2]), ylim([-3 3])
```


Img
coupe(Img)
|TF(Img)|^0.5
Passe-haut
|TF(Passe-haut)|
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5
x
y
f_x
f_y

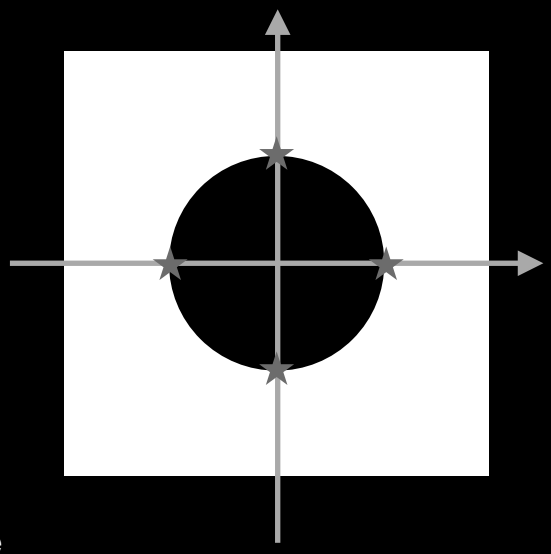
Filtrage passe-haut, passe-bande/coupe-bande, dérivateur — directement dans le plan de Fourier

L'avantage d'être dans le plan de Fourier, c'est que l'on peut modeler les fréquences comme on le désire… Y compris le faire de manière plus simple.

- *Cas du filtrage passe-haut* : créer $\hat{h}$ tel que $|\hat{h}|$ croît quand la fréquence (spatiale) croît également => on favorise les fréquences les plus élevées => effet contraire du filtre passe-bas.

Application : Atténuer les variations lentes de l'image, pour faire ressortir les variations brusques (contours).

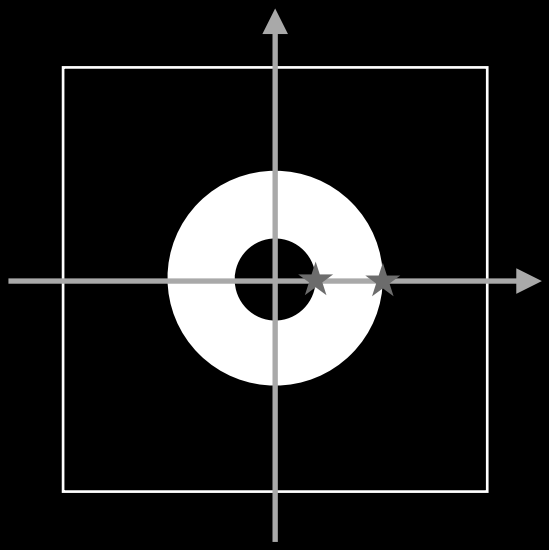
Exemple :



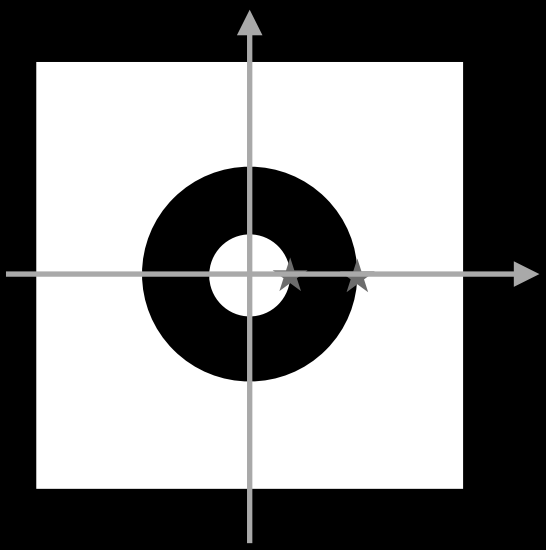
★ fréq. de coupure

(il s'agit bien d'un filtrage directement dans le plan de Fourier, appliqué à la TF de l'image)

- *<u>Cas du filtrage passe-bande :</u>* on laisse une bande de fréquences entre deux fréquences de coupures.



- *<u>Cas du filtrage coupe-bande :</u>*



- *Filtre dérivateur :*

En fait un filtre passe-haut, construit à partir de :

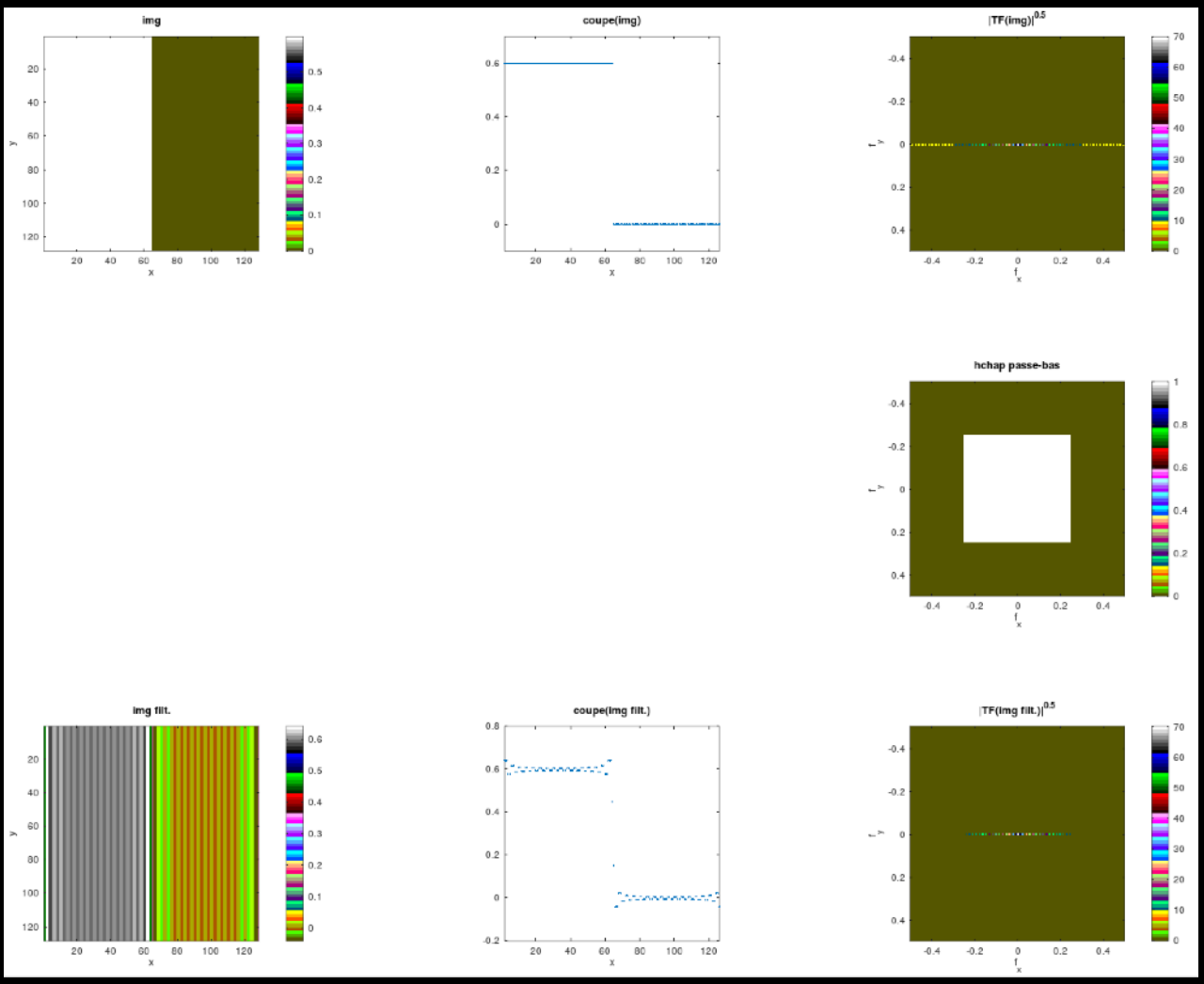
$f'(x,y) \xrightarrow{TF} 2 i \pi u \;\; 2 i \pi v \;\; \hat{f}(u,v)$

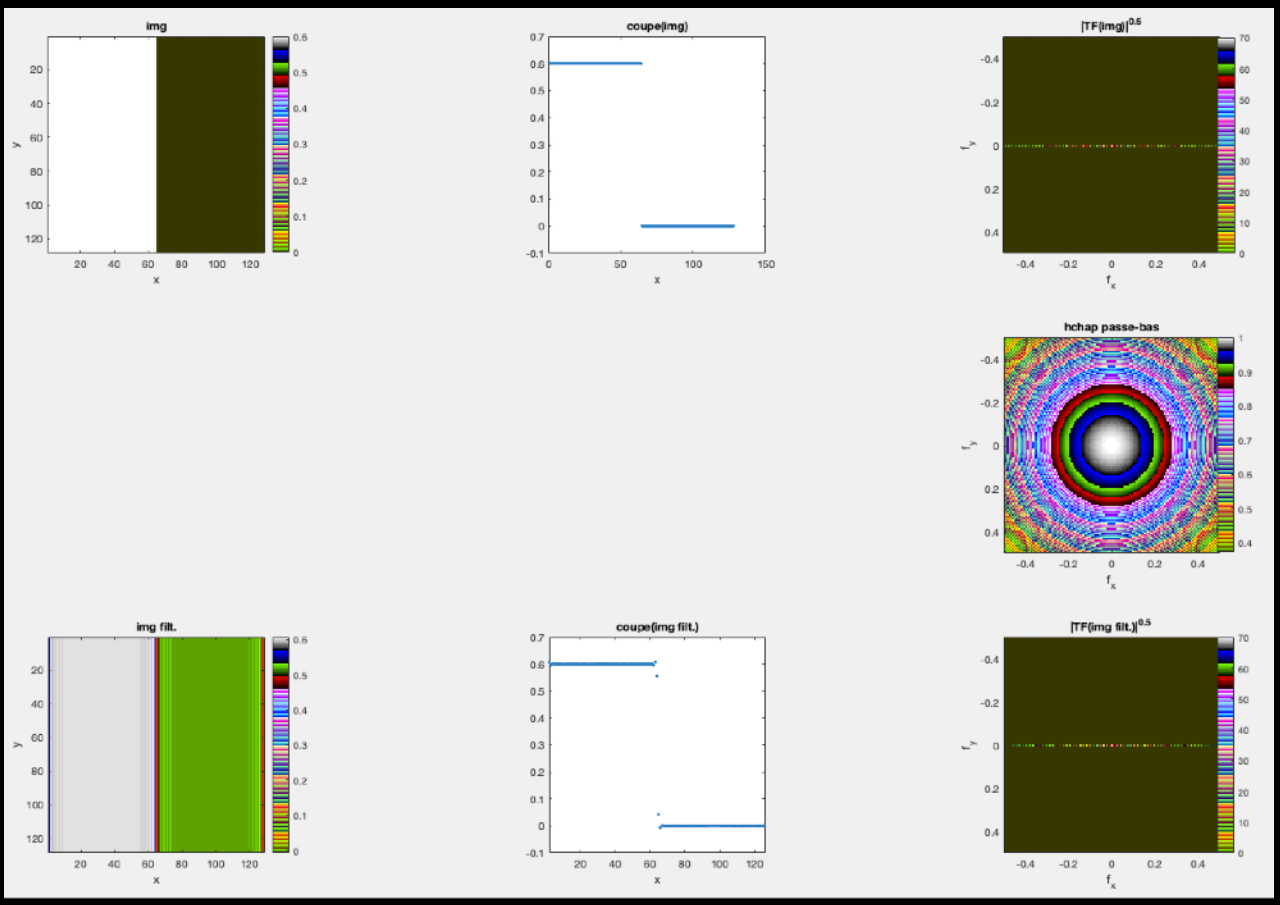
Exercice 8 :

Filtrer (en masquant dans le plan de Fourier : notion de fonction de transfert) avec un filtre passe-bas, carré, de largeur dim/2. Commenter le résultat.

```
clear
close all
%pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;        % marche 0.6/0
fx=((0:dim-1)-dim/2)/dim; fy=fx;                    % fréquences

figure, colormap('colorcube')                       % colormapp "colorcube"

subplot(3,3,1), imagesc(I)
colorbar, axis('square')
title('img'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(dim/2,:),'.')
axis('square'), ylim([-.1 .7])
title('coupe(img)'), xlabel('x')

% FFT(image)
Ichap=fft2(I,dim,dim);
Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5)
colorbar, axis('square')
title('|TF(img)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% "filtre" (en fait fct de transfert directement!)
hchap=zeros(dim,dim); nn=dim/2;
hchap(dim/2-nn/2+1:dim/2+nn/2,dim/2-nn/2+1:dim/2+nn/2)=1.;
% plus smooth : le filtre gaussien...
% hchap=fspecial('gaussian',dim,nn); hchap=hchap/max(max(hchap));
```

```
subplot(3,3,6), imagesc(fx,fy,hchap)
colorbar, axis('square')
title('hchap passe-bas'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5)
colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour dans l'espace réel
Ifilt=ifft2(Ichapfilt);
subplot(3,3,7), imagesc(real(Ifilt))
colorbar, axis('square')
title('img filt.'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(real(Ifilt(64,:)),'.')
axis('square'), ylim([-.1 .7])
title('coupe(img filt.)'), xlim([2,dim-2]), xlabel('x')
```


img
coupe(img)
|TF(img)|^0.5
hchap passe-bas
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5

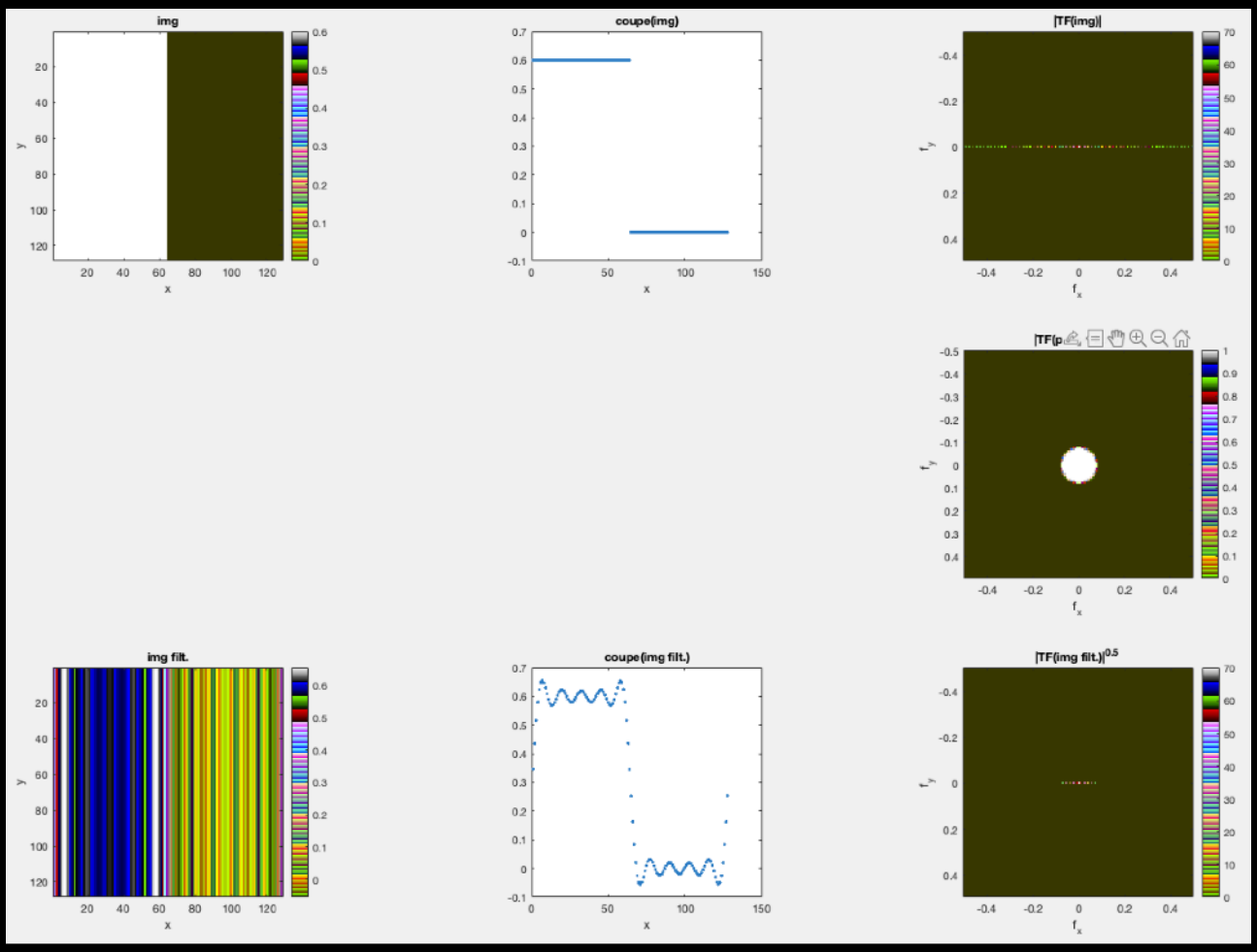

img
coupe(img)
|TF(img)|^0.5
hchap passe-bas
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5

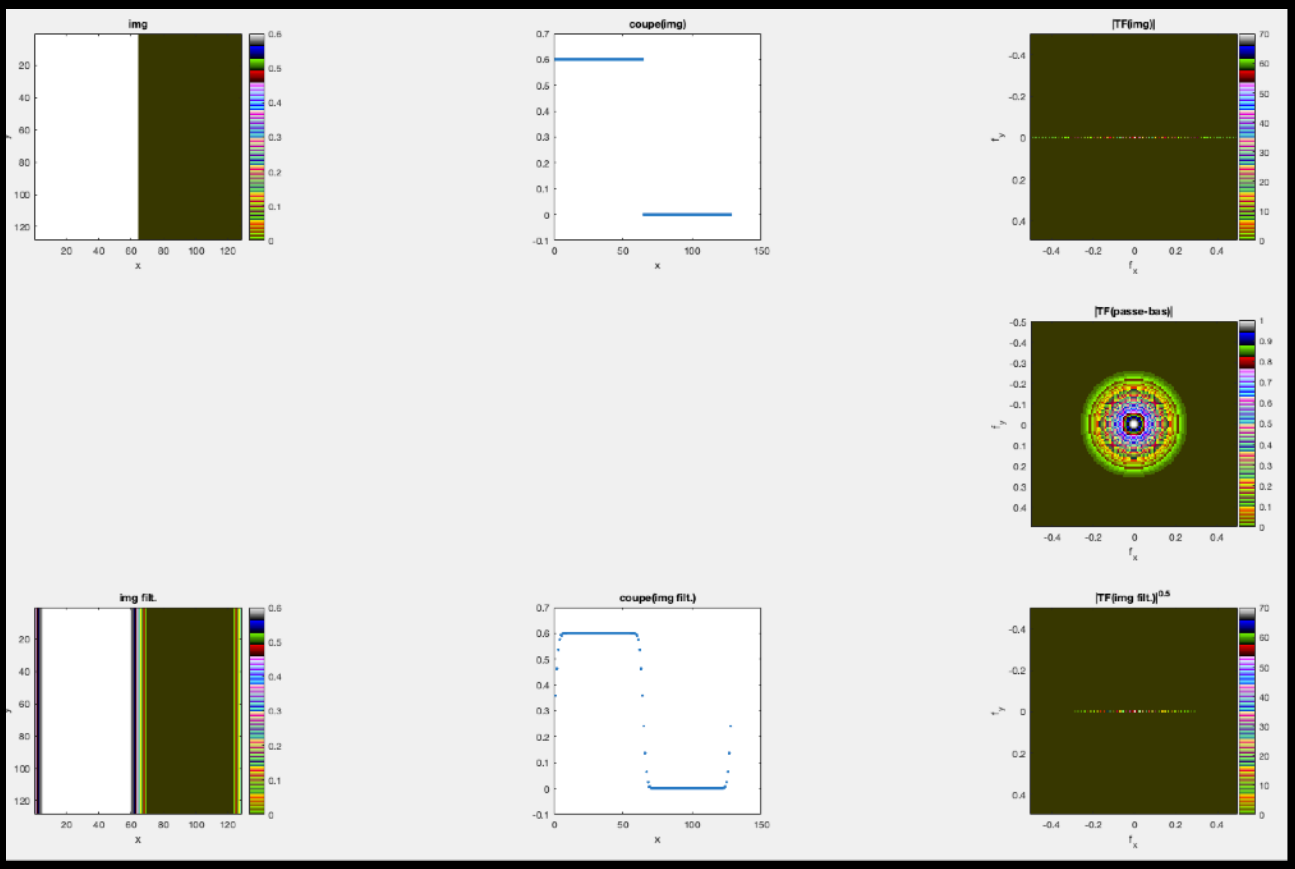
Exercice 8bis :

Même chose avec un filtre (en fait sa FFT, i.e. un masque) circulaire (10 frequels de rayon)…

```
clear
close all
%pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('colorcube')

subplot(3,3,1), imagesc(I)
colorbar,axis('square')
title('img'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(64,:),'.'), axis('square'), ylim([-.1 .7])
title('coupe(img)'), xlabel('x')

% FFT(image)
Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5)
colorbar, axis('square')
title('|TF(img)|'), xlabel('f_x'), ylabel('f_y')

% "filtre" (en fait fct de transfert directement!)
hchap=zeros(dim,dim); rr=10;
% hh=fspecial('disk',rr); hh=hh/max(max(hh));
% hchap(dim/2+1-rr:dim/2+1+rr,dim/2+1-rr:dim/2+1+rr)=hh;
% plus smooth : le filtre gaussien...
hchap=fspecial('gaussian',dim,rr); hchap=hchap/max(max(hchap));
```

```
% filtrage dans l'espace de Fourier
Ichapfilt=Ichap.*fftshift(hchap);
Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5)
colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour dans l'espace réel
Ifilt=ifft2(Ichapfilt);
subplot(3,3,7), imagesc(real(Ifilt))
colorbar, axis('square')
title('img filt.'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(real(Ifilt(64,:)),'.'), axis('square'), ylim([-.1 .7])
title('coupe(img filt.)'), xlabel('x')
```


img
coupe(img)
|TF(img)|
|TF(p
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5
x
y
f_x
f_y

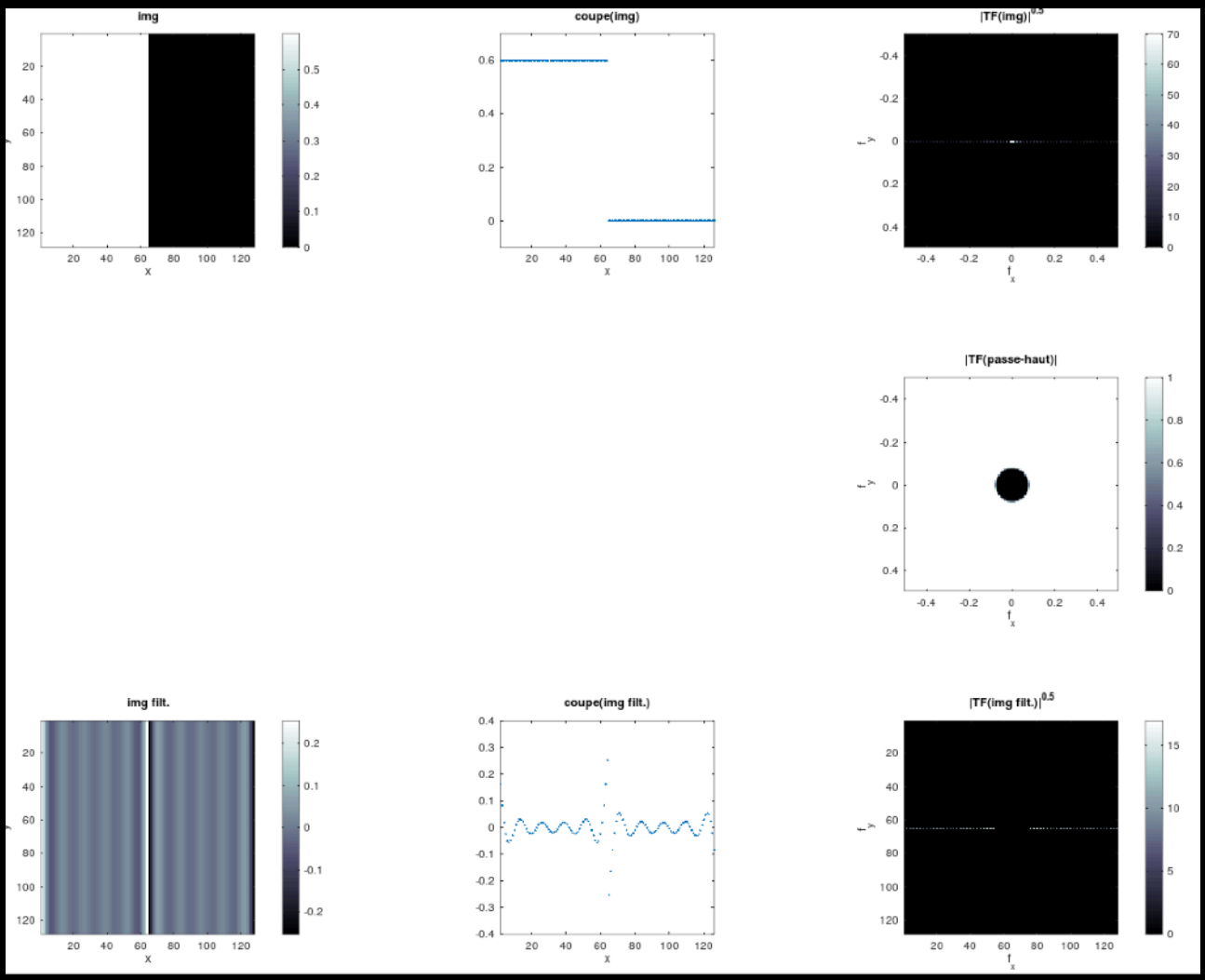

img
coupe(img)
|TF(img)|
|TF(passe-bas)|
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5
x
y
f_x
f_y

Exercice 9 :

Faire de même avec un filtre passe-haut, circulaire de rayon 10 frequels. Commenter le résultat.

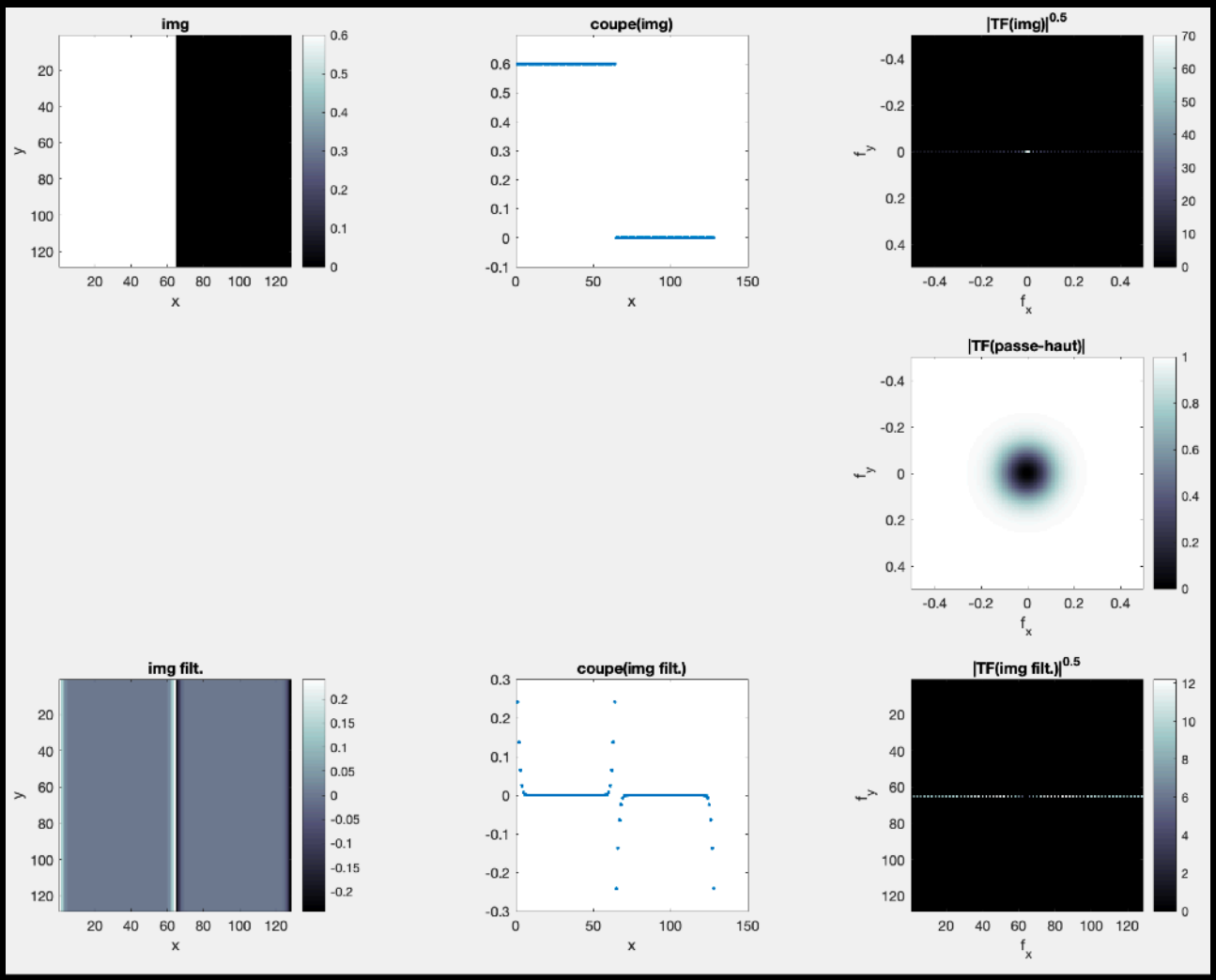
```
clear
close all
%pkg load image

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('bone')
subplot(3,3,1), imagesc(I)
colorbar, axis('square')
title('img'), xlabel('x'), ylabel('y')

subplot(3,3,2), plot(I(64,:),'.')
axis('square'), ylim([-.1 .7])
title('coupe(img)'), xlabel('x')

% FFT(image)
Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5), colorbar, axis('square')
title('|TF(img)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% "filtre" (en fait fct de transfert directement!)
hchap=ones(dim,dim);
rr=10; hh=fspecial('disk',rr); hh=hh/max(max(hh));
hchap(dim/2+1-rr:dim/2+1+rr,dim/2+1-rr:dim/2+1+rr)=1.-hh;

subplot(3,3,6), imagesc(fx,fy,hchap), colorbar, axis('square')
title('|TF(passe-haut)|'), xlabel('f_x'), ylabel('f_y')
```

```
% filtrage dans l'espace de Fourier
Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(Ichapfiltmod.^.5)
colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour dans l'espace réel
Ifilt=ifft2(Ichapfilt);
subplot(3,3,7), imagesc(real(Ifilt)), colorbar, axis('square')
title('img filt.'), xlabel('x'), ylabel('y')

subplot(3,3,8), plot(real(Ifilt(64,:)),'.')
axis('square'), title('coupe(img filt.)'), xlabel('x')
```



<u>Exercice 9+ :</u> Même chose (ou presque) mais avec moins d'oscillations ?…

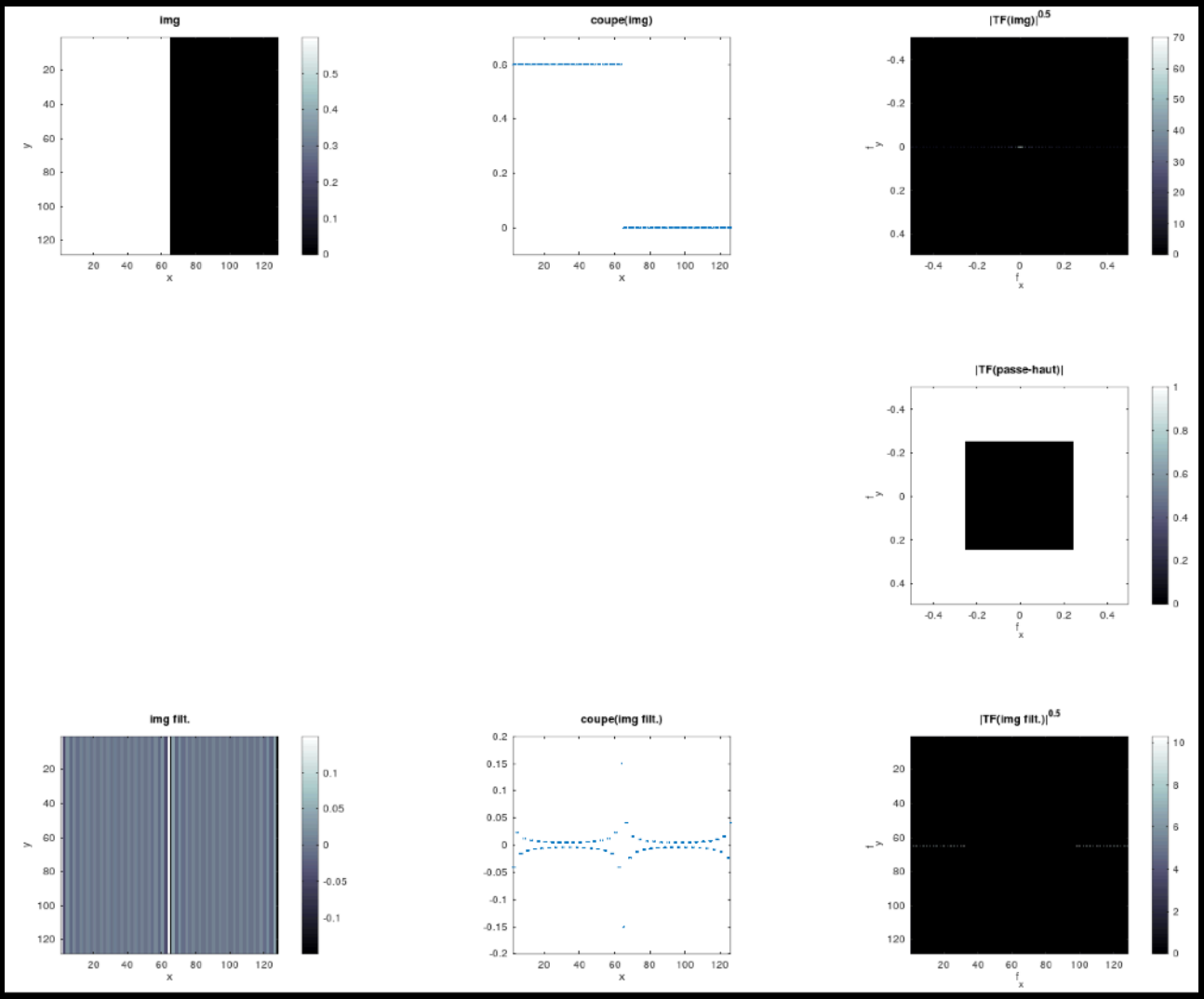
```
24  % "filtre" (en fait fct de transfert directement!)
25  hchap=ones(dim,dim); rr=10;
26  % hh=fspecial('disk',rr); hh=hh/max(max(hh));
27  % hchap(dim/2+1-rr:dim/2+1+rr,dim/2+1-rr:dim/2+1+rr)=1.-hh;
28  % plus smooth : un filtre avec une forme gaussienne...
29  hchap=fspecial('gaussian',dim,rr); hchap=hchap/max(max(hchap));
30  hchap=1-hchap;
```



Exercice 9bis :

Même chose avec un masque carré de dimension dim/2.

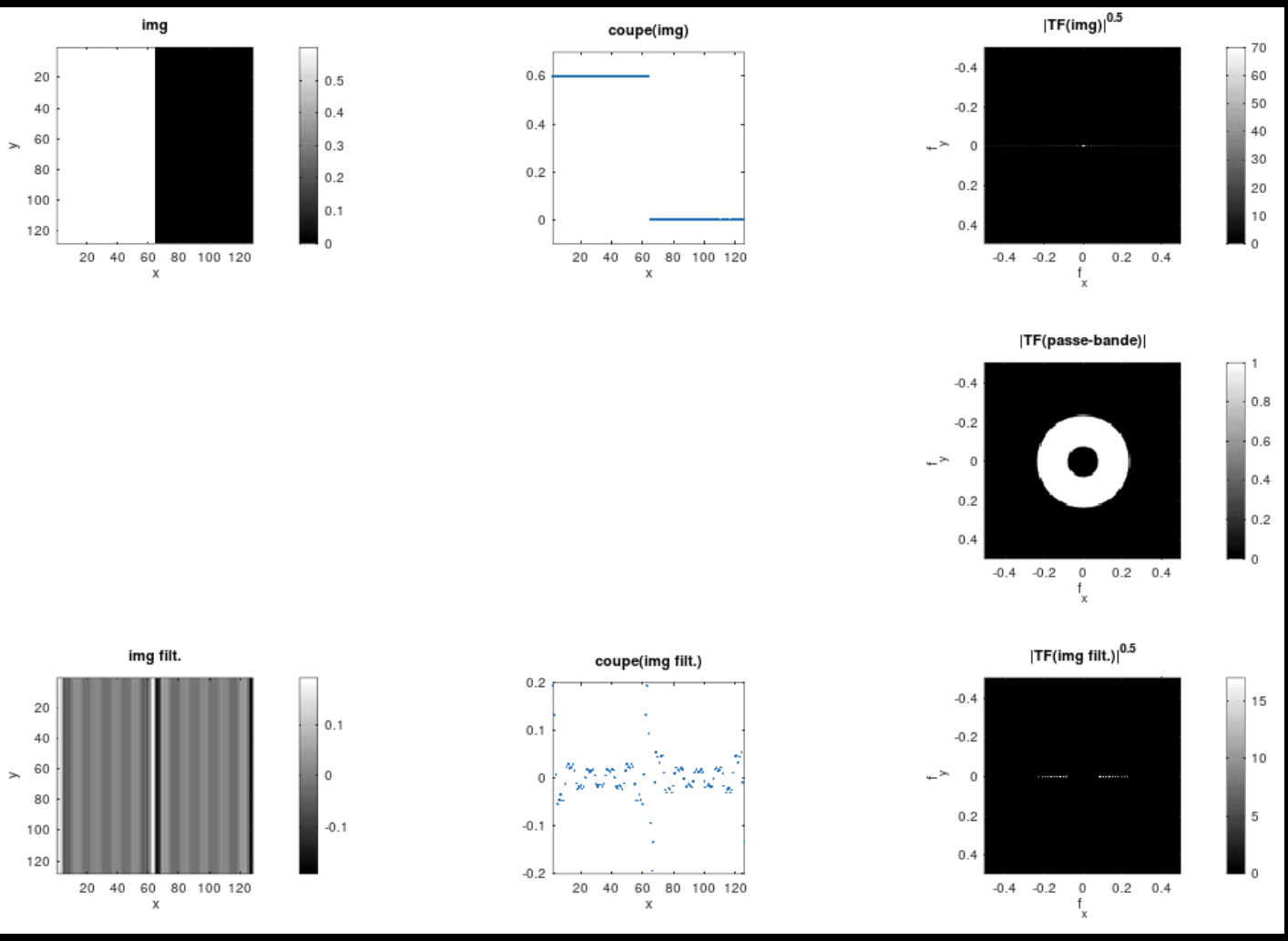
```
clear
close all

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('bone')
subplot(3,3,1), imagesc(I), colorbar, axis('square'), title('I(x,y)')

subplot(3,3,2), plot(I(64,:),'.'), axis('square'), ylim([-.1 .7])
title('coupe(img)'), xlabel('x')

% FFT(image)
Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5), colorbar, axis('square')
title('|TF(img)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% masque dans le plan de Fourier (fct de transfert)
hchap=zeros(dim,dim); nn=dim/2;
hchap(dim/2-nn/2+1:dim/2+nn/2,dim/2-nn/2+1:dim/2+nn/2)=1.; hchap=1-hchap;
subplot(3,3,6), imagesc(fx,fy,hchap), colorbar, axis('square')
title('|TF(passe-haut)|'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));
subplot(3,3,9), imagesc(Ichapfiltmod.^.5), colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour dans l'espace réel
Ifilt=real(ifft2(Ichapfilt));
subplot(3,3,7), imagesc(Ifilt), colorbar, axis('square'), title('I_f(x,y)')

subplot(3,3,8), plot(real(Ifilt(64,:)),'.'), axis('square')
title('coupe(img filt.)'), xlabel('x')
```


img
coupe(img)
|TF(img)|^0.5
|TF(passe-haut)|
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5
x
y
f_x
f_y

Exercice 10 : Idem avec un filtre passe-bande circulaire entre 10 et 30 frequels.

```
clear
close all

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('gray')
subplot(3,3,1), imagesc(I), colorbar, axis('square'), title('I(x,y)')
subplot(3,3,2), plot(I(64,:),'.'), axis('square'), title('coupe de I')
ylim([-.1 .7]), xlabel('x')

% FFT(image)
Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));
subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5), colorbar, axis('square')
title('|TF(img)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% masque dans le plan de Fourier (fct de transfert)
hchap=zeros(dim,dim);
rr =10; hh =fspecial('disk',rr ); hh =hh /max(max(hh)) ;
rrr=30; hhh=fspecial('disk',rrr); hhh=hhh/max(max(hhh));
hchap(dim/2+1-rrr:dim/2+1+rrr,dim/2+1-rrr:dim/2+1+rrr)=hhh;
hchap(dim/2+1-rr :dim/2+1+rr ,dim/2+1-rr :dim/2+1+rr )=1-hh;
```

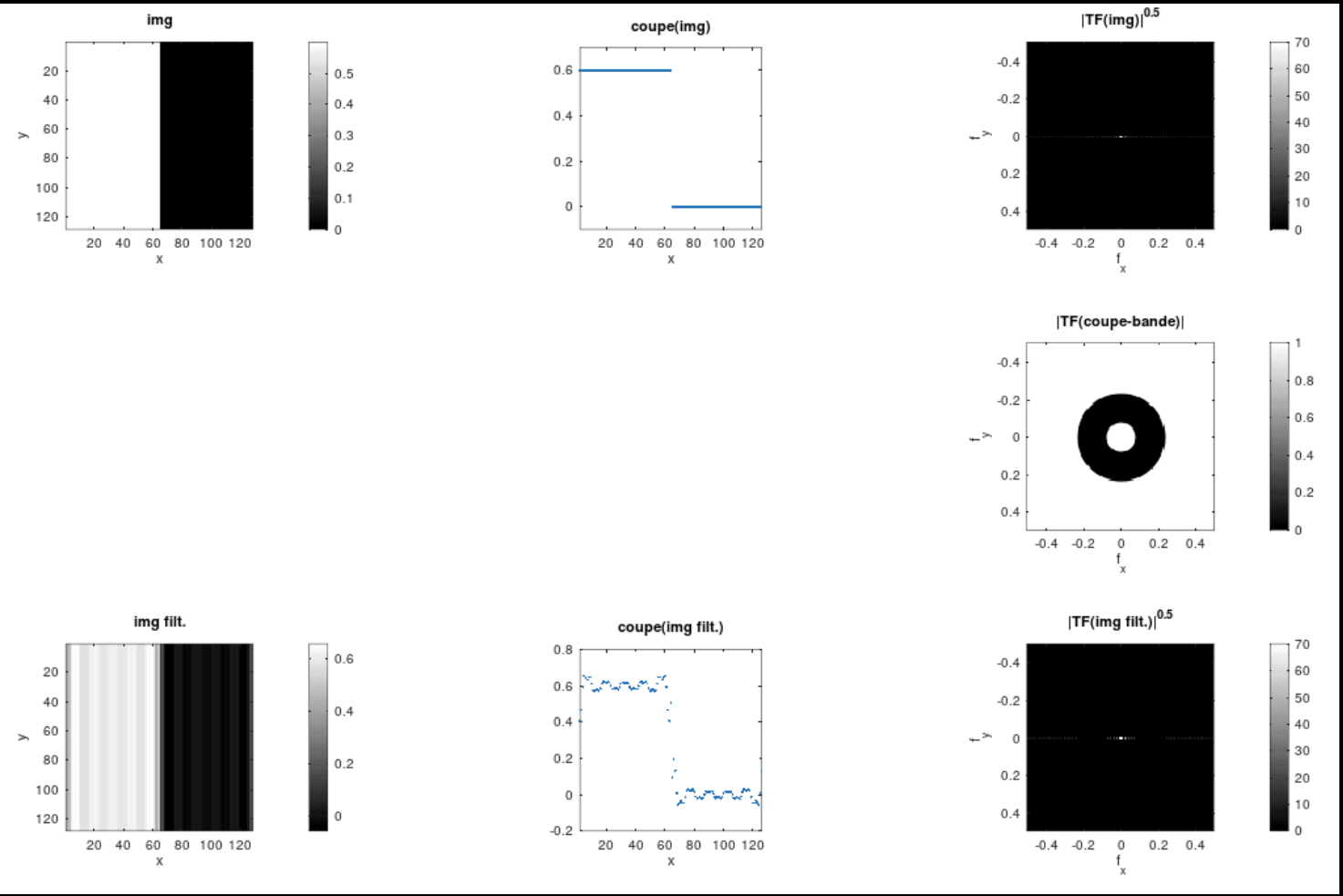
```
subplot(3,3,6), imagesc(fx,fy,hchap), colorbar, axis('square')
title('|TF(passe-bande)|'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));

subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5), colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour dans l'espace réel
Ifilt=real(ifft2(Ichapfilt));

subplot(3,3,7), imagesc(Ifilt), colorbar, axis('square'), title('I_f(x,y)')
subplot(3,3,8), plot(Ifilt(64,:),'.'), axis('square')
title('coupe de I_f'), xlabel('x')
```


img
coupe(img)
|TF(img)|^0.5
|TF(passe-bande)|
img filt.
coupe(img filt.)
|TF(img filt.)|^0.5
x
y
f_x
f_y

Exercice 11 : Idem avec un filtre coupe-bande circulaire entre 10 et 30 frequels.

```
clear
close all

% image
dim=128; I=zeros(dim,dim); I(:,1:dim/2)=0.6;
fx=((0:dim-1)-dim/2)/dim; fy=fx;

figure, colormap('gray')
subplot(3,3,1), imagesc(I), colorbar, axis('square'), title('I(x,y)')

subplot(3,3,2), plot(I(dim/2,:),'.'), axis('square'), title('coupe de I')
ylim([-.1 .7]), xlabel('x')

% FFT(image)
Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));

subplot(3,3,3), imagesc(fx,fy,Ichapmod.^.5), colorbar, axis('square')
title('|TF(img)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% "filtre" (en fait fct de transfert directement!)
hchap=zeros(dim,dim);
rr=10; hh=fspecial('disk',rr); hh=hh/max(max(hh));
rrr=30; hhh=fspecial('disk',rrr); hhh=hhh/max(max(hhh));
hchap(dim/2+1-rrr:dim/2+1+rrr,dim/2+1-rrr:dim/2+1+rrr)=hhh;
hchap(dim/2+1-rr:dim/2+1+rr,dim/2+1-rr:dim/2+1+rr)=1-hh;
hchap=1.-hchap;
```

```
subplot(3,3,6), imagesc(fx,fy,hchap), colorbar, axis('square')
title('|TF(coupe-bande)|'), xlabel('f_x'), ylabel('f_y')

% filtrage dans l'espace de Fourier
Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));

subplot(3,3,9), imagesc(fx,fy,Ichapfiltmod.^.5), colorbar, axis('square')
title('|TF(img filt.)|^{0.5}'), xlabel('f_x'), ylabel('f_y')

% retour dans l'espace réel
Ifilt=real(ifft2(Ichapfilt));

subplot(3,3,7), imagesc(Ifilt), colorbar, axis('square'), title('I_f(x,y)')
subplot(3,3,8), plot(Ifilt(dim/2,:),'.'), axis('square')
title('coupede I_f'), xlabel('x')
```



—> Boîte à outils pour le filtrage de Fourier !!

Etape 1 : Création de la fonction
Dans un fichier nommé stat.m, on écrit le texte suivant :

Code :

```
1  function [mean,stdev] = stat(x)
2  n = length(x);
3  mean = sum(x)/n;
4  stdev = sqrt(sum((x-mean).^2/n));
```

Etape 2 : Création d'un script qui utilise la fonction :
Dans un fichier nommé monscript.m (qui se trouve dans l

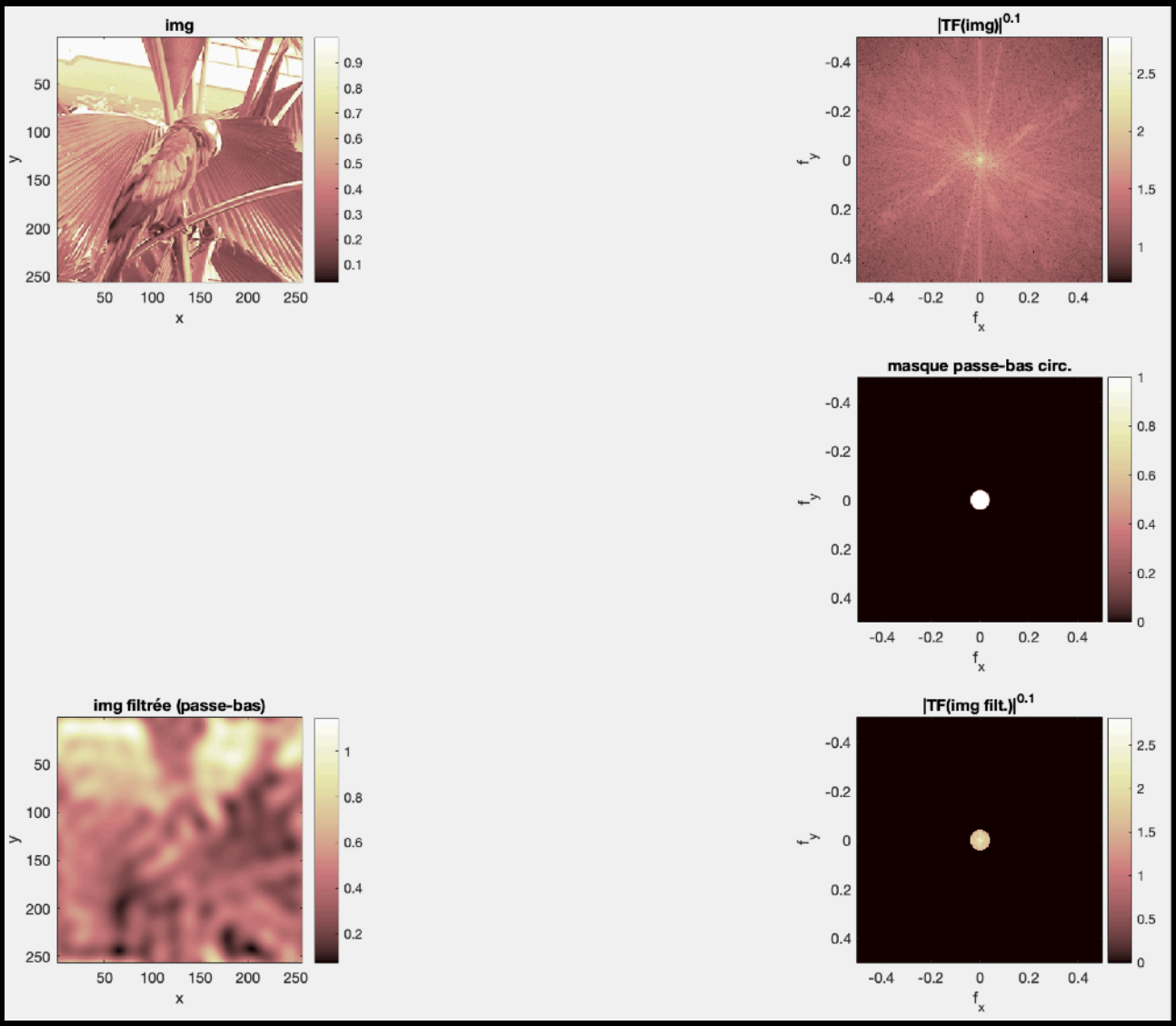
Code :

```
[mean stdev] = stat([12.7 45.4 98.9 26.6 53/1])
```

Exercice 12 : Appliquer les filtrages passe-bas circulaire, passe-haut circulaire, passe-bande circulaire et coupe-bande circulaire sur une image de votre choix. (Vérifier au passage que la somme de l'image filtrée passe-bas et de l'image filtrée passe-haut redonne bien l'image de départ. Idem pour les filtres passe-bande et coupe-bande.)

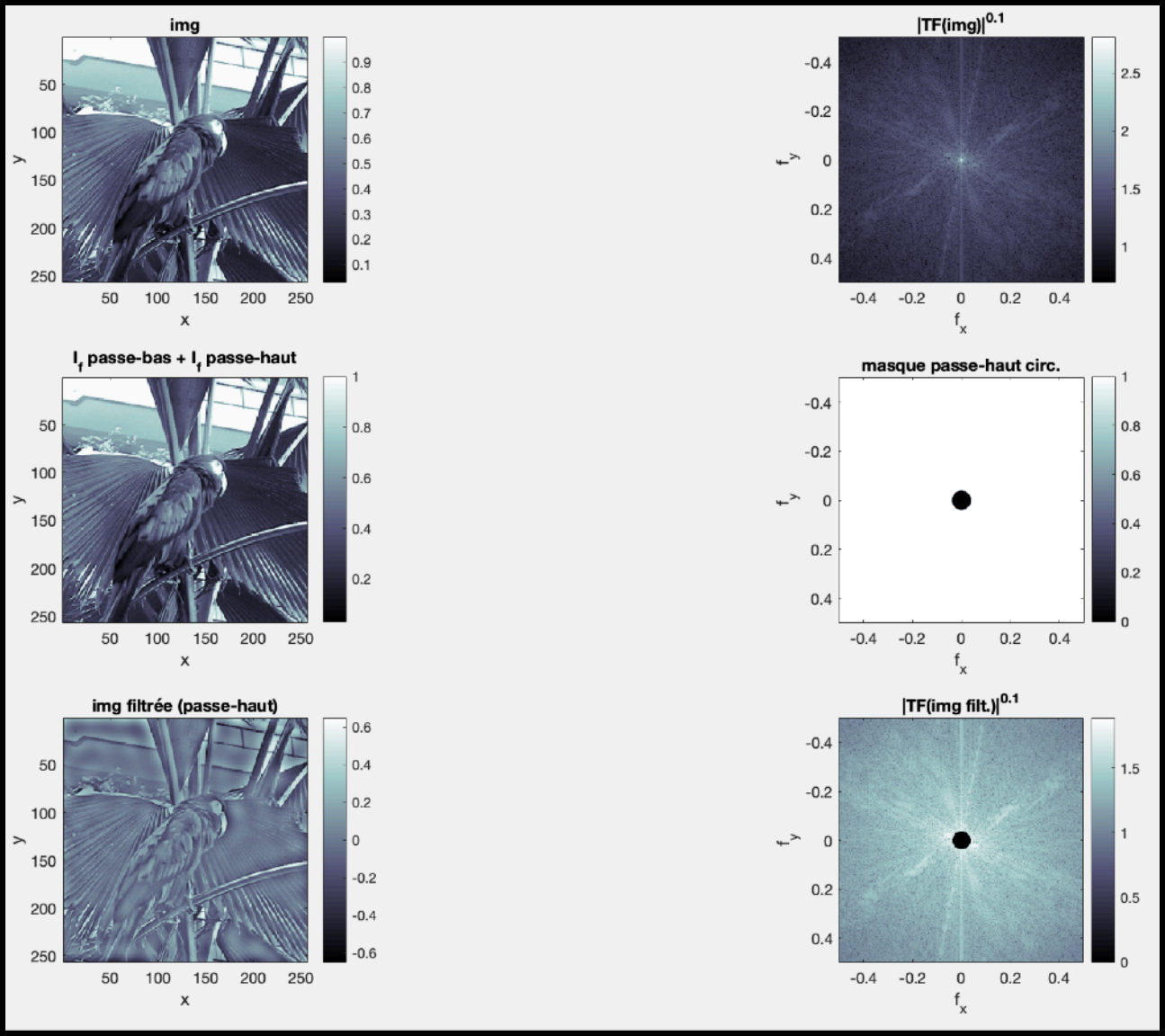
```
clear
close all
%pkg load image

% image et FFT(image)
I=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/bird.jpg');
I=rgb2gray(I);
I=double(I)/255;
dimx=size(I,1); dimy=size(I,2);

fx=((0:dimx-1)-dimx/2)/(dimx*1); % ici Δx=Δy=1
fy=((0:dimy-1)-dimy/2)/(dimy*1);

Ichap=fft2(I);
Ichapmod=abs(fftshift(Ichap));

% (1) filtrage passe-bas circulaire
rr=10; hh=fspecial('disk',rr); hh=hh/max(max(hh));
hchap=zeros(dimx,dimy);
hchap(dimx/2+1-rr:dimx/2+1+rr,dimy/2+1-rr:dimy/2+1+rr)=hh;

Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));

Ifilt=real(ifft2(Ichapfilt));
```

```
% représentation
figure(1), colormap('pink')

subplot(3,2,1), imagesc(I), colorbar, axis('image')
title('img'), xlabel('x'), ylabel('y')

subplot(3,2,2), imagesc(fx,fy,Ichapmod.^.1), colorbar, axis('image'),
title('|TF(img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,4), imagesc(fx,fy,hchap), colorbar, axis('image')
title('masque passe-bas circ.'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,6),imagesc(fx,fy,Ichapfiltmod.^.1), colorbar, axis('image')
title('|TF(img filt.)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,5), imagesc(Ifilt), colorbar, axis('image')
title('img filtrée (passe-bas)'), xlabel('x'), ylabel('y')
```

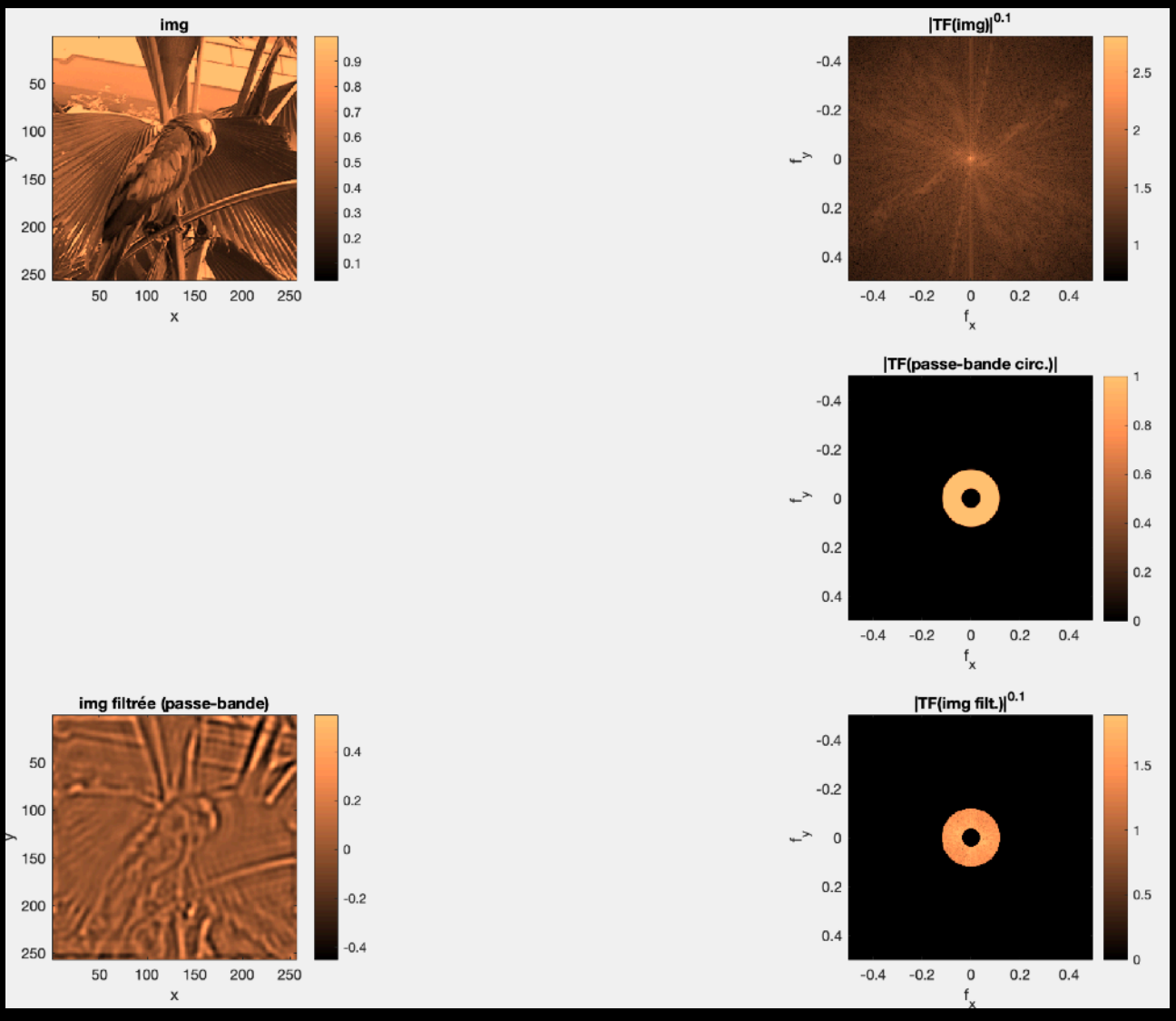

img
|TF(img)|^0.1
masque passe-bas circ.
img filtrée (passe-bas)
|TF(img filt.)|^0.1
x
y
f_x
f_y

```
% (2) filtrage passe-haut circulaire
hchap=ones(dimx,dimy);
hchap(dimx/2+1-rr:dimx/2+1+rr,dimy/2+1-rr:dimy/2+1+rr)=1.-hh;

Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));

Ifilt2=real(ifft2(Ichapfilt));

% représentation
figure(2), colormap('bone')

subplot(3,2,1), imagesc(I), colorbar, axis('image')
title('img'), xlabel('x'), ylabel('y')

subplot(3,2,2), imagesc(fx,fy,Ichapmod.^.1), colorbar, axis('image')
title('|TF(img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,4), imagesc(fx,fy,hchap), colorbar, axis('image')
title('masque passe-haut circ.'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,6),imagesc(fx,fy,Ichapfiltmod.^.1), colorbar, axis('image')
title('|TF(img filt.)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,5), imagesc(Ifilt2), colorbar, axis('image')
title('img filtrée (passe-haut)'), xlabel('x'), ylabel('y')
```

```
% somme des deux images filtrées = image de départ ?
Iff=Ifilt+Ifilt2;
subplot(3,2,3), imagesc(Iff), colorbar, axis('image')
title('I_f passe-bas + I_f passe-haut'), xlabel('x'), ylabel('y')
distance=sqrt(sum(sum((I-Iff).^2)))/(dimx*dimy);
['distance entre image de départ et somme des deux images filtrées : ', num2str(distance)]
['comparée à l"intégrale de l"image : ', num2str(sum(sum(I)))]
['soit ', num2str(distance/sum(sum(I))*100), '% de différence']
```

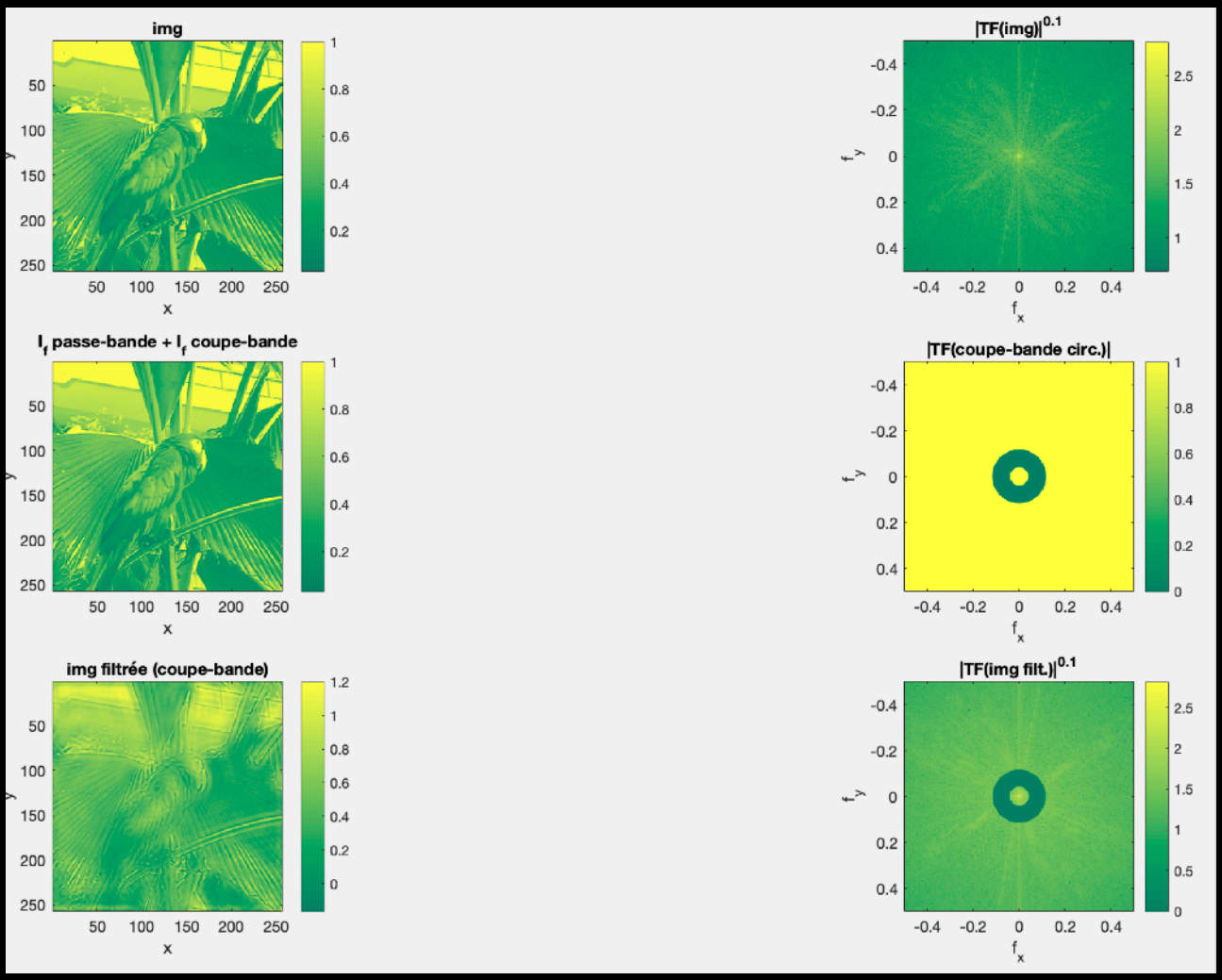

img
|TF(img)|^0.1
I_f passe-bas + I_f passe-haut
masque passe-haut circ.
img filtrée (passe-haut)
|TF(img filt.)|^0.1
x
y
f_x
f_y

```
% (3) filtrage passe-bande circulaire
hchap=zeros(dimx,dimy);
rrr=30; hhh=fspecial('disk',rrr); hhh=hhh/max(max(hhh));
hchap(dimx/2+1-rrr:dimx/2+1+rrr,dimy/2+1-rrr:dimy/2+1+rrr)=hhh;
hchap(dimx/2+1-rr :dimx/2+1+rr ,dimy/2+1-rr :dimy/2+1+rr )=1-hh;

Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));

Ifilt=real(ifft2(Ichapfilt));

% représentation
figure(3), colormap('copper')

subplot(3,2,1), imagesc(I), colorbar, axis('image')
title('img'), xlabel('x'), ylabel('y')

subplot(3,2,2), imagesc(fx,fy,Ichapmod.^.1), colorbar, axis('image')
title('|TF(img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,4), imagesc(fx,fy,hchap), colorbar, axis('image')
title('|TF(passe-bande circ.)|'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,6),imagesc(fx,fy,Ichapfiltmod.^.1), colorbar, axis('image')
title('|TF(img filt.)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,5), imagesc(Ifilt), colorbar, axis('image')
title('img filtrée (passe-bande)'), xlabel('x'), ylabel('y')
```



```
% (4) filtrage coupe-bande circulaire
hchap=1.-hchap;

Ichapfilt=fftshift(hchap).*Ichap;
Ichapfiltmod=abs(fftshift(Ichapfilt));

Ifilt2=real(ifft2(Ichapfilt));

% représentation
figure(4), colormap('summer')

subplot(3,2,1), imagesc(I), colorbar, axis('image')
title('img'), xlabel('x'), ylabel('y')

subplot(3,2,2), imagesc(fx,fy,Ichapmod.^.1), colorbar, axis('image')
title('|TF(img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,4), imagesc(fx,fy,hchap), colorbar, axis('image')
title('|TF(coupe-bande circ.)|'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,6),imagesc(fx,fy,Ichapfiltmod.^.1), colorbar, axis('image')
title('|TF(img filt.)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,5), imagesc(Ifilt2), colorbar, axis('image')
title('img filtrée (coupe-bande)'), xlabel('x'), ylabel('y')
```

```
% somme des deux images filtrées = image de départ ?
Iff=Ifilt+Ifilt2;
subplot(3,2,3), imagesc(Iff), colorbar, axis('image')
title('I_f passe-bande + I_f coupe-bande'), xlabel('x'), ylabel('y')
distance=sqrt(sum(sum((I-Iff).^2)))/(dimx*dimy);
['distance entre image de départ et somme des deux images filtrées : ', num2str(distance)]
['comparée à l"intégrale de l"image : ', num2str(sum(sum(I)))]
['soit ', num2str(distance/sum(sum(I))*100), '% de différence']
```

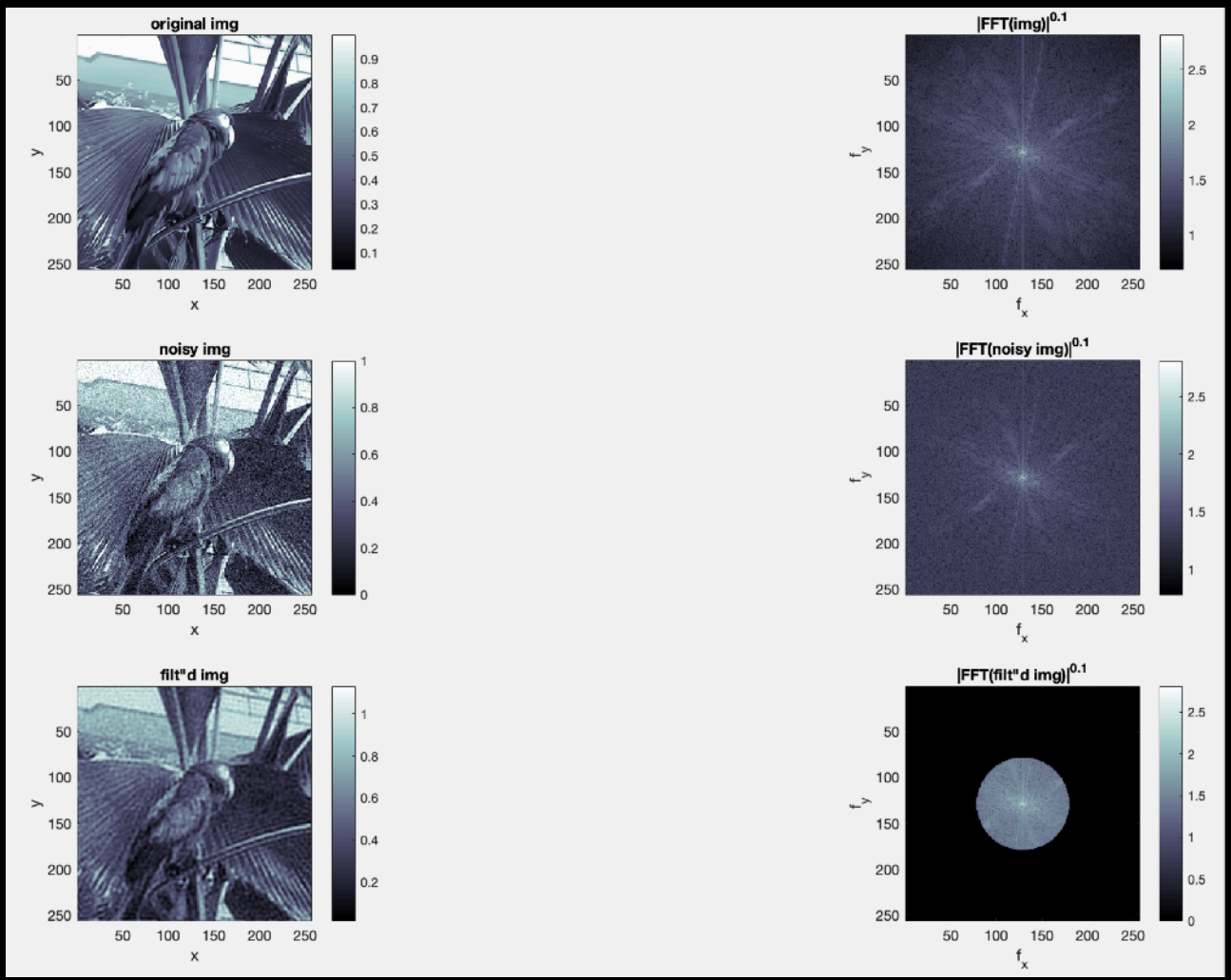


<u>Exercice 12bis :</u> Filtrer une image de votre choix par un filtre qui ne laisse passer qu'une partie des fréquences (passe-bande). Prenez une bande de fréquences chacun·e puis vérifier que l'on retrouve bien l'image initiale en additionnant toutes les images filtrées en une image composite.

3 - Restoration d'images

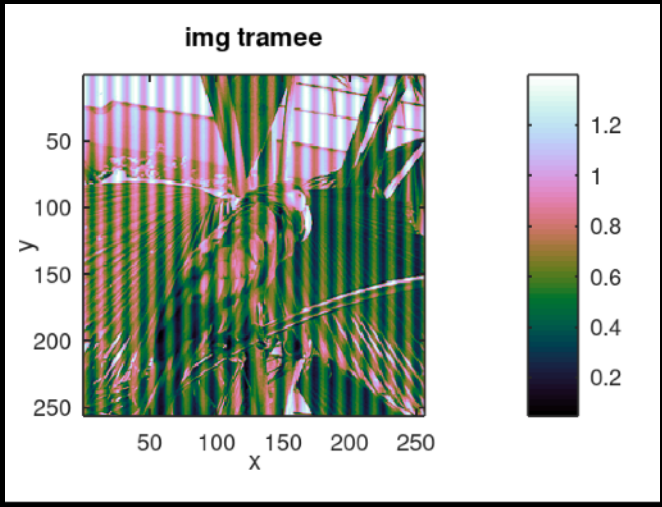
- **Exercice 13 :** Bruiter (bruit Gaussien additif de variance 1%) une image de votre choix (e.g. *bird.jpeg*), puis atténuer ce bruit à l'aide d'un filtre passe-bas adapté par vos soins. Quelle est la fréquence de coupure optimale pour ce filtre ?

```
%%% préliminaires
clear
close all
%pkg load image

%%% préparation image
img=imread('../0-images/bird.jpg');
img=rgb2gray(img); img=double(img)/255.;
dim=size(img); dimx=dim(1); dimy=dim(2);
fx=((0:dimx-1)-dimx/2)/dimx;
fy=((0:dimy-1)-dimy/2)/dimy;

%%% FFT(image) - pour comparaison
img_chap = fft2(img);
img_chapmod = abs(fftshift(img_chap));

%%% figure
figure, colormap('bone')

subplot(3,2,1)
imagesc(img), colorbar, axis('image')
title('original img'), xlabel('x'), ylabel('y')

subplot(3,2,2)
imagesc(img_chapmod.^.1), colorbar, axis('image')
title('|FFT(img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

%%% bruitage
varg = 0.01;
imgn = imnoise(img,'gaussian',0.,varg);

%%% FFT(image bruitée)
imgn_chap=fft2(imgn);
imgn_chapmod=abs(fftshift(imgn_chap));

subplot(3,2,3)
imagesc(imgn), colorbar, axis('image')
title('noisy img'), xlabel('x'), ylabel('y')

subplot(3,2,4)
imagesc(imgn_chapmod.^.1), colorbar, axis('image')
title('|FFT(noisy img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

%%% filtrage passe-bas (débruitage)
hchap=zeros(dim);
rr=50; hh=fspecial('disk',rr); hh=hh/max(max(hh));
hchap(dimx/2+1-rr:dimx/2+1+rr,dimy/2+1-rr:dimy/2+1+rr)=hh;

imgn_chap_filt=fftshift(hchap).*imgn_chap;
imgn_chap_filtmod=abs(fftshift(imgn_chap_filt));
imgn_filt=real(ifft2(imgn_chap_filt));

subplot(3,2,5)
imagesc(imgn_filt), colorbar, axis('image')
title('filt"d img'), xlabel('x'), ylabel('y')

subplot(3,2,6)
imagesc(imgn_chap_filtmod.^.1), colorbar, axis('image')
title('|FFT(filt"d img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')
```


original img
|FFT(img)|^0.1
noisy img
|FFT(noisy img)|^0.1
filt"d img
|FFT(filt"d img)|^0.1
x
y
f_x
f_y

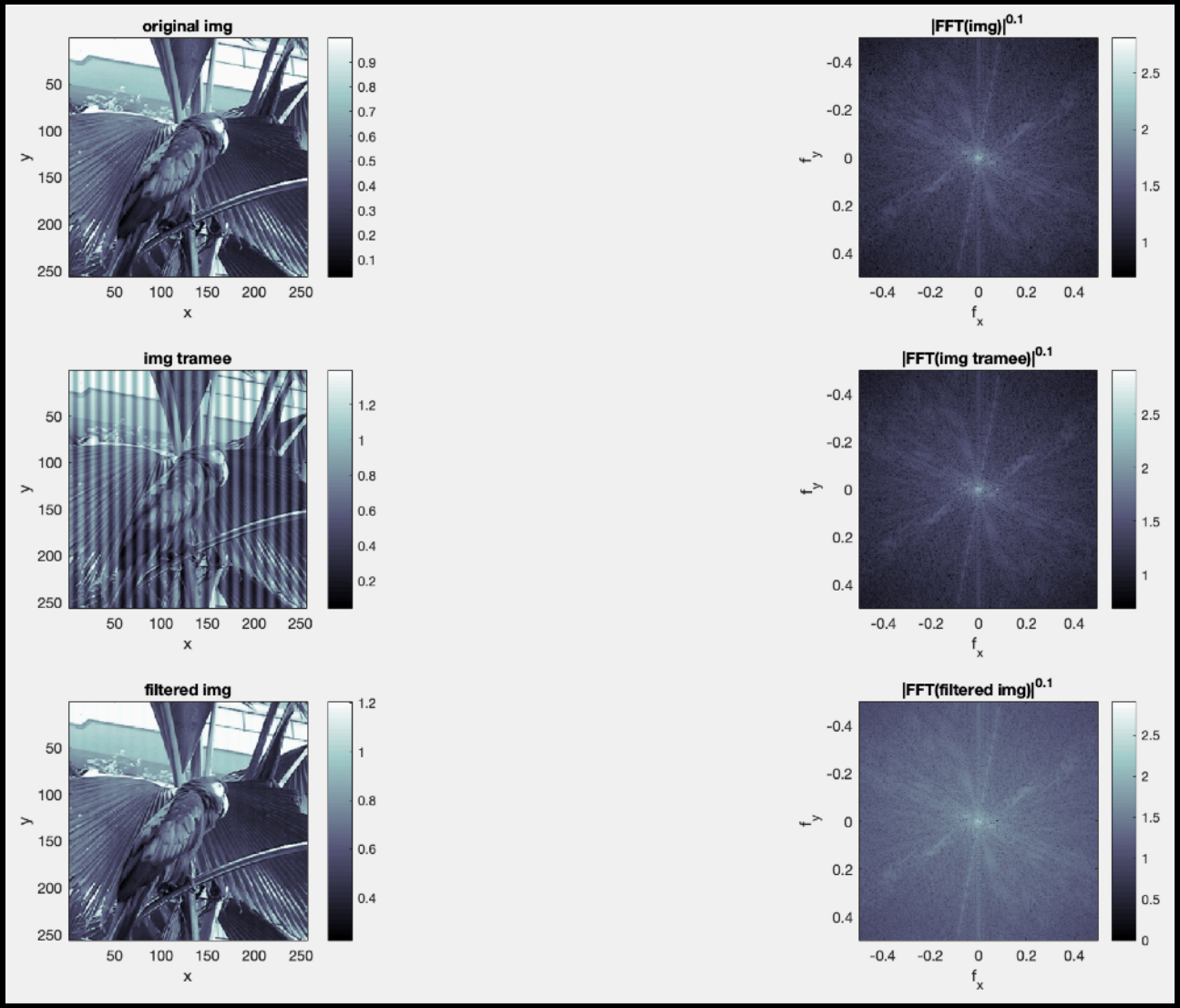
- **<u>Exercice 14 : Détramage</u>**
  Tramer (avec une haute fréquence), puis atténuer le tramage dans l'image de votre choix (ou a priori *bird.jpeg*) en filtrant dans le plan de Fourier.

[Tramage avec un cosinus tel que :
*tramage = 1 + cos(2πx/T)*coeff, T=12.8 px, coeff=0.2,*
puis faire : *image = image+tramage*]




```
%%% (0) PRÉLIMINAIRES

%%% préliminaires
clear
close all
%pkg load image

%%% préparation image
img=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/bird.jpg');
img=rgb2gray(img); img=double(img)/255.;
dim=size(img); dim=dim(1);
fx=((0:dim-1)-dim/2)/dim; fy=((0:dim-1)-dim/2)/dim;

%%% FFT(image) - pour comparaison
img_chap = fftshift(fft2(img)); % on se place dans un plan de Fourier
                                % cette fois-ci ré-ordonné !

%%% (1) TRAMAGE

Tx=12.8;                        % Tx=12.8 [px]
                                % => fc [en px^-1] = 1/Tx ≈ 0.08 px^(-1)
                                % => fc [en frequels] = 1/Tx*dim = 20 frequels
x=0:(dim-1); coeff=.2;
tram=ones(dim,1)*(1+cos(2*pi*x/Tx))*coeff;
imgt=img+tram;
imgt_chap = fftshift(fft2(imgt));
```

```
%%% (2) DÉTRAMAGE

%%% filtrage coupe-fréquel (détramage)
imgt_chap_filt=imgt_chap;
fc = dim/Tx;
imgt_chap_filt(dim/2+1,dim/2+1+fc)=0.;
imgt_chap_filt(dim/2+1,dim/2+1-fc)=0.;
imgt_filt=real(ifft2(ifftshift(imgt_chap_filt)));
                                % à cause du ré-ordonnage dans le plan de
                                % Fourier, on est obligé d'utiliser
                                % ifftshift avant de faire la FFT inverse

%%% résultat tramage/détramage
figure, colormap('bone')

subplot(3,2,1)
imagesc(img), colorbar, axis('square')
title('original img'), xlabel('x'), ylabel('y')

subplot(3,2,2)
imagesc(fx,fy,abs(img_chap).^.1), colorbar, axis('square')
title('|FFT(img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,3)
imagesc(imgt), colorbar, axis('square')
title('img tramee'), xlabel('x'), ylabel('y')

subplot(3,2,4)
imagesc(fx,fy,abs(imgt_chap).^.1), colorbar, axis('square')
title('|FFT(img tramee)|^{0.1}'), xlabel('f_x'), ylabel('f_y')

subplot(3,2,5)
imagesc(imgt_filt), colorbar, axis('square')
title('filtered img'), xlabel('x'), ylabel('y')

subplot(3,2,6)
imagesc(fx,fy,abs(imgt_chap_filt).^.1), colorbar, axis('square')
title('|FFT(filtered img)|^{0.1}'), xlabel('f_x'), ylabel('f_y')
```


original img
|FFT(img)|^0.1
img tramee
|FFT(img tramee)|^0.1
filtered img
|FFT(filtered img)|^0.1
x
y
f_x
f_y

4 - Reconstruction d'image

Il s'agit de « déflouter » (en fait ***déconvoluer***) une image, rendue floue par ailleurs.

$$I_{\text{floue}} = I \otimes G$$

où $\otimes$ est l'opérateur de la ***convolution*** et G la fonction qui rend floue l'image I.

$$\Rightarrow \hat{I}_{\text{floue}} = \hat{I} \times \hat{G}$$

où **x** est ici une multiplication élément par élément.

Une idée (qui est naturelle mais pas forcément très bonne en présence de bruit, ne serait-ce que numérique) serait de poser :

$$\hat{I} = \hat{I}_{\text{floue}} \; / \; \hat{G}$$

Mais on doit déjà considérer le fait que l'on est en présence de nombres complexes, et donc on a :

$$\hat{I} = |\hat{I}| \; \exp\{i\phi_I\}$$

$$\hat{G} = |\hat{G}| \; \exp\{i\phi_G\}$$

$$\hat{I}_{\text{floue}} = |\hat{I}_{\text{floue}}| \; \exp\{i\phi_{I_{\text{floue}}}\}$$

d'où :

$$\hat{I} = \frac{|\hat{I}_{\text{floue}}|}{|\hat{G}|} \exp\left(\iota \left(\phi_{I_{\text{floue}}} - \phi_G\right)\right)$$

où la division des deux modules est a priori très délicate à cause des valeurs de $|\hat{G}|$ quand elles sont proches de zéro.

=> solution : appliquer un seuil à $|\hat{G}|$ afin d'éviter les points qui vont diverger vers l'infini à cause de la division…

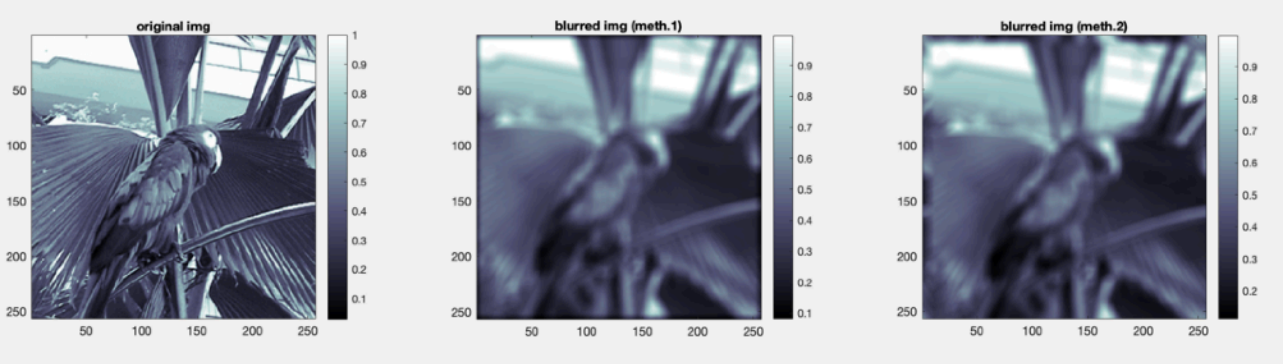
… et ce en appliquant :

- $|\hat{G}| \geq s$: $|\hat{G}|$ ok $\Rightarrow \hat{G}$ ok.
- $|\hat{G}| < s$: $|\hat{G}| = \text{seuil} \Rightarrow \hat{G} = \text{seuil} \times \exp\{\iota\phi_G\}$

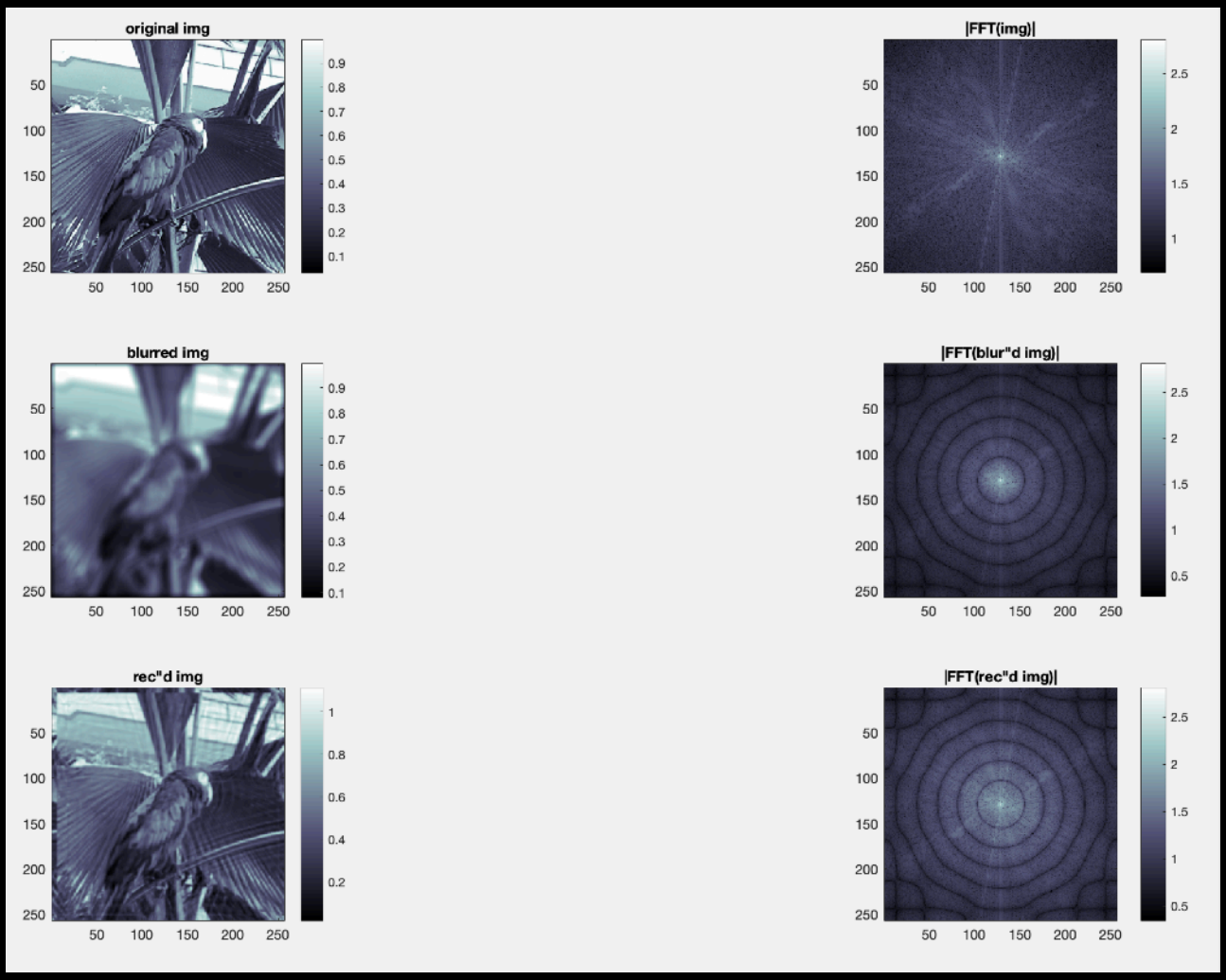
Il s'agit d'un ***filtre inverse***.

- **Exercice 15 :** Prendre l'image précédente, la flouter en la convoluant par un disque de rayon 3, puis 6, puis 12px, puis appliquer un *filtre inverse* afin d'opérer une tentative de reconstruction de l'image initiale. Jouer avec le seuil.

```
%%%%%%%%%%%%%%%%%%%%%%%%
%%% FILTRAGE INVERSE %%%
%%%%%%%%%%%%%%%%%%%%%%%%

%%% préliminaires
clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PREMIÈRE PARTIE : SIMULATION %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% préparation image
img=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/bird.jpg');
img=rgb2gray(img); img=double(img)/255.;
dim=size(img); dimx=dim(1); dimy=dim(2);
img_chap = fft2(img);   % pour comparaison

%%% floutage (convolution par un disque)
rr=3; gag=fspecial('disk',rr);
%%% convolution dans le plan direct
gau=zeros(dim);
gau(dimx/2+1-rr:dimx/2+1+rr, dimy/2+1-rr:dimy/2+1+rr)=gag;
imgg=convn(img,gau,'same');
%%% solution alternative (dans le plan de Fourier)
gag_chap=fft2(gag,dimx,dimy);
imgg_chap=gag_chap.*img_chap;
imgg2 = real(ifft2(imgg_chap));
%%% affichage floutage
figure(1), colormap('bone')
subplot(1,3,1), imagesc(img), colorbar
title('original img'), axis('image')
subplot(1,3,2), imagesc(imgg), colorbar
title('blurred img (meth.1)'), axis('image')
subplot(1,3,3), imagesc(imgg2), colorbar
title('blurred img (meth.2)'), axis('image')
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% DEUXIÈME PARTIE : TRAITEMENT %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% défloutage par filtrage inverse
imgg_chap=fft2(imgg2);
gau_chap=fft2(gau);
seuil = .01;   % on peut jouer ici avec le seuil pour un résultat optimal
idx=find(abs(gau_chap) < seuil);
gau_chap(idx)=seuil*exp(complex(0,1)*angle(gau_chap(idx)));
img_rec_chap=imgg_chap./gau_chap;
img_rec=real(ifftshift(ifft2(img_rec_chap)));

%%% affichage filtrage inverse
figure(2), colormap('bone')
subplot(3,2,1), imagesc(img), colorbar
title('original img'), axis('image')
subplot(3,2,2), imagesc(abs(fftshift(img_chap)).^.1), colorbar
title('|FFT(img)|'), axis('image')
subplot(3,2,3), imagesc(imgg), colorbar
title('blurred image'), axis('image')
subplot(3,2,4), imagesc(abs(fftshift(imgg_chap)).^.1), colorbar
title('|FFT(blurred img)|'), axis('image')
subplot(3,2,5), imagesc(img_rec), colorbar
title("rec'd image"), axis('image')
subplot(3,2,6), imagesc(abs(fftshift(img_rec_chap)).^.1), colorbar
title("|FFT(rec'd image)|"), axis('image')
```

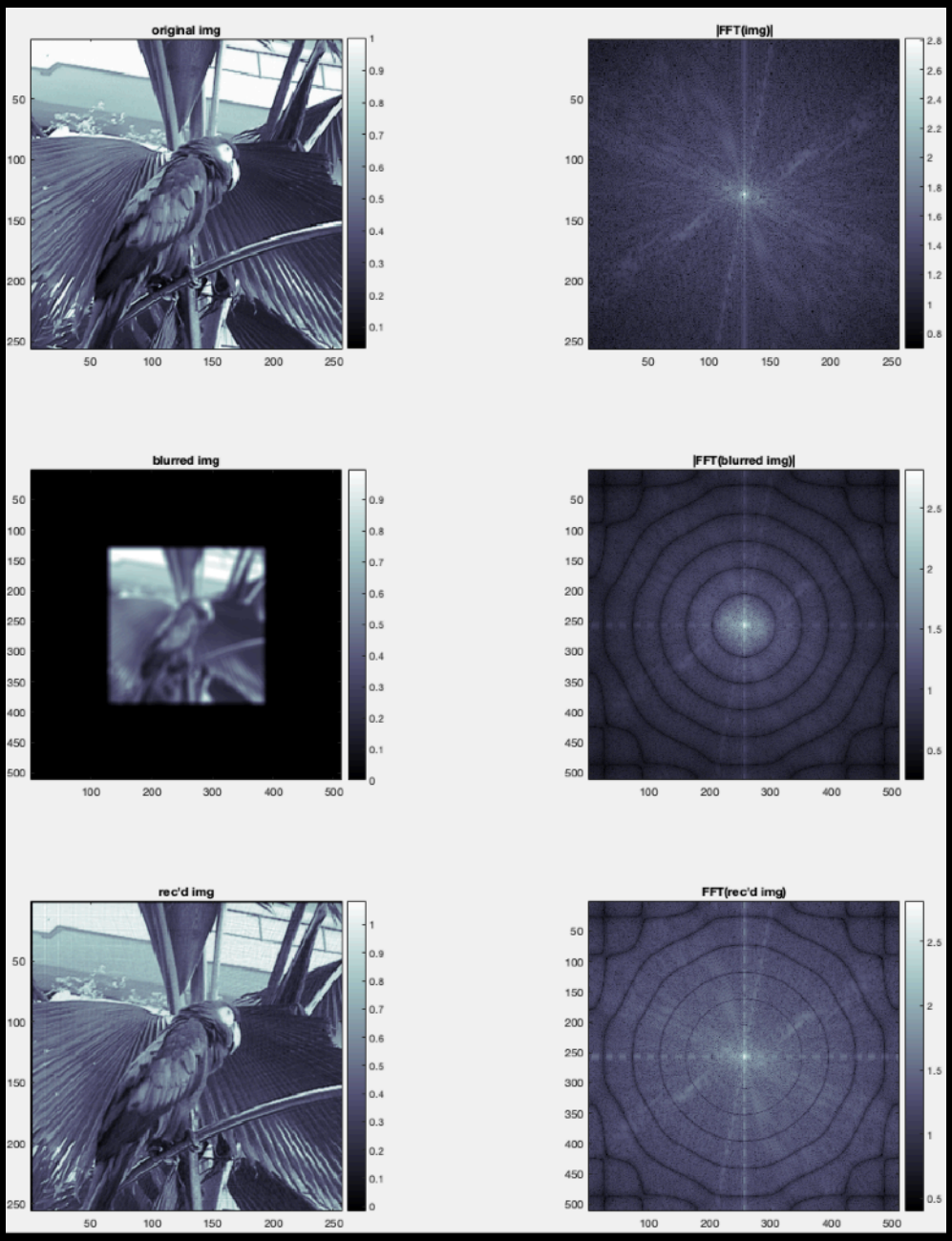


—> on remarque des artefacts dus aux bords…
=> solution : zero-padding !

- **Exercice 16 :** Reprendre l'exercice 15 en appliquant du pavage de zéro (zero-padding) dans des tableaux deux fois plus grands (linéairement, i.e. quatre fois plus grands en surface!).

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% FILTRAGE INVERSE AVEC ZEZRO_PADDING %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% préliminaires
clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PREMIÈRE PARTIE : SIMULATION %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% préparation image
img=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/bird.jpg');
img=rgb2gray(img); img=double(img)/255.;
dim=size(img); dimx=dim(1); dimy=dim(2);
img_chap = fft2(img,2*dimx,2*dimy);  % FFT (+zero-padding au passage)

%%% floutage (convolution par un disque) (+zero-padding au passage)
rr=6; gag=fspecial('disk',rr);
gau=zeros(dim); gau(dimx/2+1-rr:dimx/2+1+rr,dimy/2+1-rr:dimy/2+1+rr)=gag;
dummy=conv2(img,gau,'full'); imgg=zeros(2*dimx,2*dimy); imgg(2:end,2:end)=dummy;
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% DEUXIÈME PARTIE : TRAITEMENT %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% défloutage par filtrage inverse
gau_chap=fft2(gau,2*dimx,2*dimy);
imgg_chap=fft2(imgg);
seuil = .01;   % on peut jouer ici avec le seuil pour un résultat optimal
idx=find(abs(gau_chap) < seuil);
gau_chap(idx)=seuil*exp(complex(0,1)*angle(gau_chap(idx)));
img_rec_chap=imgg_chap./gau_chap;
img_rec=real(ifft2(img_rec_chap));
img_rec=img_rec(1:dimx,1:dimy); % pas besoin de réordonner les quadrants
                                % ici, on prend directement le bon
%%% affichage filtrage inverse avec zero padding
figure(), colormap('bone')
subplot(3,2,1), imagesc(img), colorbar
title('original img'), axis('image')
subplot(3,2,2), imagesc(abs(fftshift(fft2(img))).^.1), colorbar
title('|FFT(img)|'), axis('image')
subplot(3,2,3), imagesc(imgg), colorbar
title('blurred img'), axis('image')
subplot(3,2,4), imagesc(abs(fftshift(imgg_chap)).^.1), colorbar
title('|FFT(blurred img)|'), axis('image')
subplot(3,2,5), imagesc(img_rec), colorbar
title("rec'd img"), axis('image')
subplot(3,2,6), imagesc(abs(fftshift(img_rec_chap)).^.1), colorbar
title("FFT(rec'd img)"), axis('square')
```

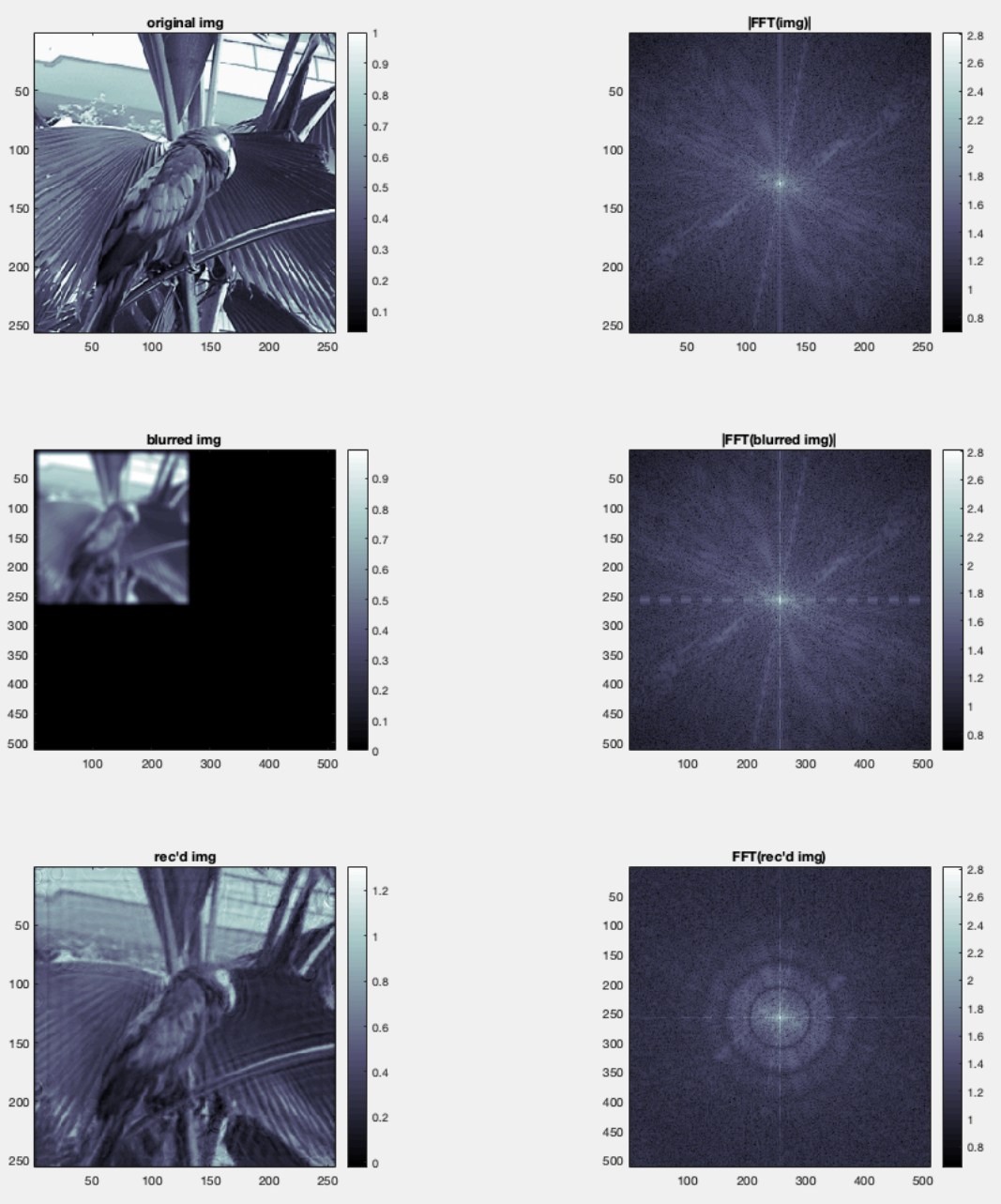


- Préférer à partir de maintenant la méthode de convolution en passant par Fourier pour le floutage.

- **Exercice 17 :** Reprendre l’exercice en lui appliquant un bruit poivre et sel affectant 2% des pixels. Après avoir tenter une reconstruction de l’image floutée par inversion, penser à tenter une réduction du bruit poivre et sel avant, puis jouer avec le seuil pour trouver un équilibre entre artefacts et propagation du bruit.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% FILTRAGE INVERSE EN PRÉSENCE DE BRUIT P&S %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% préliminaires
clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PREMIÈRE PARTIE : SIMULATION %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% préparation image
img=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/bird.jpg');
img=rgb2gray(img); img=double(img)/255.;
dim=size(img); dimx=dim(1); dimy=dim(2);
img_chap = fft2(img,2*dimx,2*dimy);  % FFT (+zero-padding au passage)

%%% floutage (convolution par un disque) avec zero-padding
%%% solution alternative (dans le plan de Fourier)
rr=6; gag=fspecial('disk',rr); gag_chap=fft2(gag,2*dimx,2*dimy);
imgg_chap=gag_chap.*img_chap; imgg = real(ifft2(imgg_chap));

%%% bruitage poivre et sel à 2%
imgg=imnoise(imgg, 'salt & pepper',0.02);
```

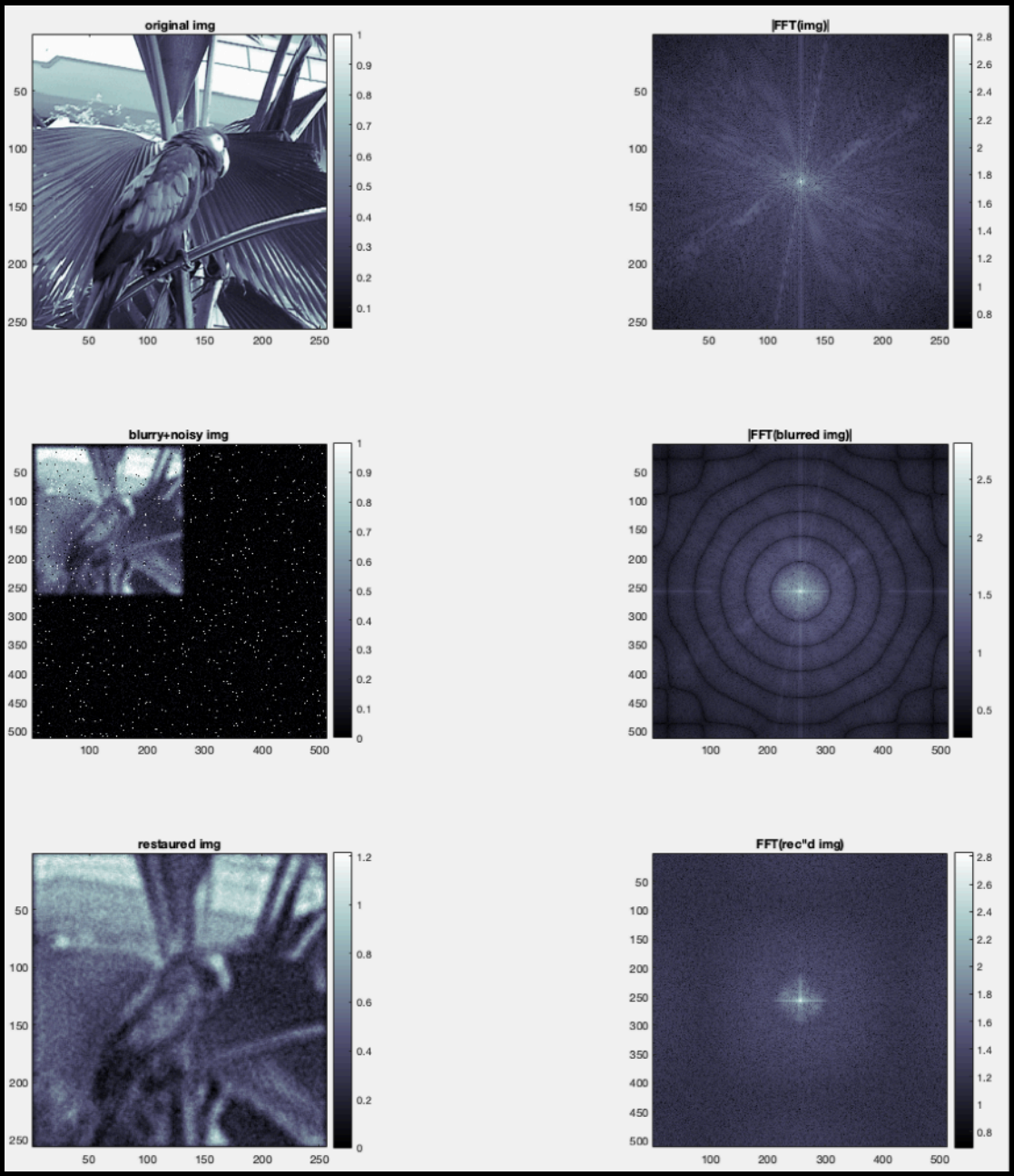
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% DEUXIÈME PARTIE : TRAITEMENT %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% filtre médian pour contrer le bruit p&s
imgg=medfilt2(imgg);

%%% défloutage par filtrage inverse
gau_chap=gag_chap;
imgg_chap=fft2(imgg);
seuil = .1;   % on a ici monté le seuil (à optimiser!)
idx=find(abs(gau_chap) < seuil);
gau_chap(idx)=seuil*exp(complex(0,1)*angle(gau_chap(idx)));
img_rec_chap=imgg_chap./gau_chap;
img_rec=real(ifft2(img_rec_chap));
img_rec=img_rec(1:dimx,1:dimy);

%%% affichage filtrage inverse avec zero padding en présence de bruit p&s
figure(), colormap('bone')
subplot(3,2,1), imagesc(img), colorbar
title('original img'), axis('image')
subplot(3,2,2), imagesc(abs(fftshift(fft2(img))).^.1), colorbar
title('|FFT(img)|'), axis('image')
subplot(3,2,3), imagesc(imgg), colorbar
title('blurred img'), axis('image')
subplot(3,2,4), imagesc(abs(fftshift(img_chap)).^.1), colorbar
title('|FFT(blurred img)|'), axis('image')
subplot(3,2,5), imagesc(img_rec), colorbar
title("rec'd img"), axis('image')
subplot(3,2,6), imagesc(abs(fftshift(img_rec_chap)).^.1), colorbar
title("FFT(rec'd img)"), axis('image')
```



- **<u>Exercice 18 :</u>** Reprendre l'exercice en lui appliquant toujours un bruit poivre et sel (2% des pixels), et de surcroît un bruit Gaussien d'écart type relatif 0.05. En plus du filtre non-linéaire permettant de réduire au maximum le bruit poivre et sel, jouer avec le seuil pour trouver un nouvel équilibre entre artefacts et propagation des bruits.

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% FILTRAGE INVERSE AVEC P&S ET RON %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% préliminaires
clear
close all

%%% préparation image
img=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/bird.jpg');
img=rgb2gray(img); img=double(img)/255.;
dim=size(img); dimx=dim(1); dimy=dim(2);
img_chap = fft2(img,2*dimx,2*dimy);  % FFT (+zero-padding au passage)

%%% floutage (convolution par un disque) (+zero-padding au passage)
rr=6; gag=fspecial('disk',rr); gag_chap=fft2(gag,2*dimx,2*dimy);
imgg_chap=gag_chap.*img_chap; imgg = real(ifft2(imgg_chap));

%%% bruitage poivre et sel à 2%
imgg=imnoise(imgg, 'salt & pepper', 0.02);

%%% bruitage gaussien additif (sigma=0.05)
imgg=imnoise(imgg, 'gaussian', 0., 0.05^2);

%%% filtre médian pour contrer le bruit p&s
imgg_mf=medfilt2(imgg);
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% DEUXIÈME PARTIE : TRAITEMENT %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% ajouter éventuellement une étape de réduction du bruit gaussien...

% défloutage par filtrage inverse
imgg_mf_chap=fft2(imgg_mf);
gau_chap=gag_chap;
seuil = .5;   % on a ici jouer avec le seuil (à optimiser!)
idx=find(abs(gau_chap) < seuil);
gau_chap(idx)=seuil*exp(complex(0,1)*angle(gau_chap(idx)));
img_rec_chap=imgg_mf_chap./gau_chap;
img_rec=real(ifft2(img_rec_chap)); img_rec=img_rec(1:dimx,1:dimy);

%%% affichage filtrage inverse avec zero padding
figure(), colormap('bone')
subplot(3,2,1), imagesc(img), colorbar
title('original img'), axis('image')
subplot(3,2,2), imagesc(abs(fftshift(fft2(img))).^.1), colorbar
title('|FFT(img)|'), axis('image')
subplot(3,2,3), imagesc(imgg), colorbar
title('blurry+noisy img'), axis('image')
subplot(3,2,4), imagesc(abs(fftshift(imgg_chap)).^.1), colorbar
title('|FFT(blurred img)|'), axis('image')
subplot(3,2,5), imagesc(img_rec), colorbar
title('restaured img'), axis('image')
subplot(3,2,6), imagesc(abs(fftshift(img_rec_chap)).^.1), colorbar
title('FFT(rec"d img)'), axis('image')
```


original img
|FFT(img)|
blurry+noisy img
|FFT(blurred img)|
restaured img
FFT(rec"d img)