$$x = U_0\, cos(2\pi f t + \phi)$$

La distinction entre le taux d'échantillonnage (dans ce cas, le nombre de points de données par seconde) et la fréquence du signal (nombre de cycles par seconde).

Vous vous demandez peut-être pourquoi j'ai supprimé le dernier point dans le temps ? C'est simplement parce que lors d'une FFT, Matlab suppose que le signal que vous lui donnez est périodique (fonction cosinus infinie), on l'arrêter juste avant le prochain cycle.

**FFT**

Ce tracé s'appelle un spectre de puissance

**Changer la fréquence du signal $f$ ?**

le pic se déplace vers la nouvelle fréquence. si la fréquence est nulle, le signal est plat (constant) et la **FFT** donne un pic à zéro.

**Modifier l'amplitude du signal $U_0$ ?**

L'amplitude du pic de **FFT** change également.

**Changer la phase du signal $\phi$ ?**

Le pic du spectre de puissance **ne bouge pas** dans ce cas.

**Modifier le taux d'échantillonnage $\tau$ ?**

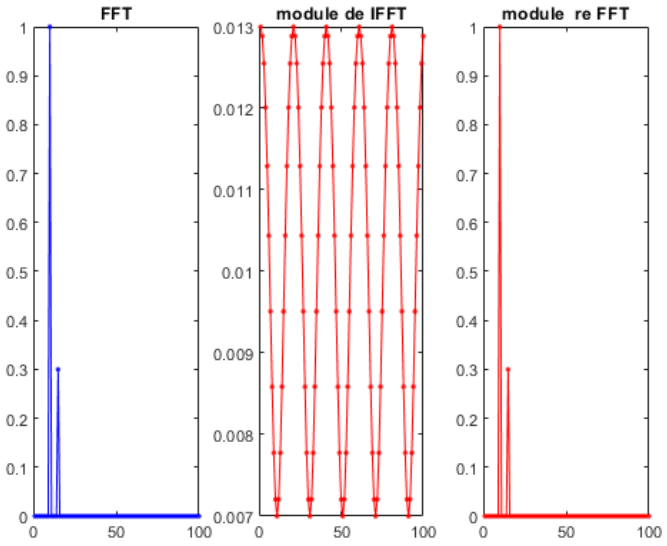
Plus le taux d'échantillonnage est élevé, plus il y a d'échantillons dans la FFT.

Si $\boldsymbol{\tau < 2f}$ le spectre de puissance commence à montrer un comportement bizarre. C'est ce qu'on appelle l'aliasing et s'explique par le théorème de Nyquist.

**Fonction delta**

On a deux pic a $t_1$ et $t_2$

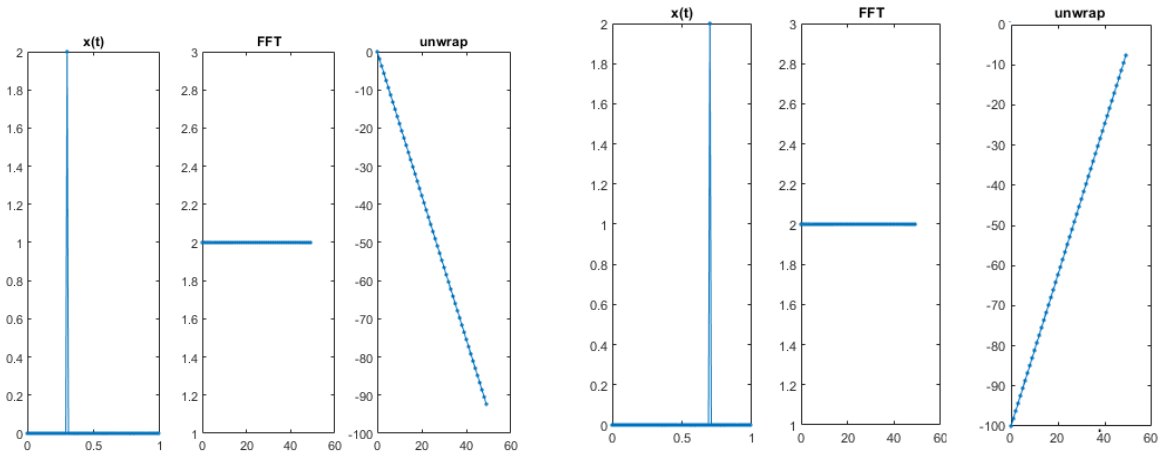
$$y = y_0 e^{i\phi} \qquad FFT = \begin{cases} e^{i\phi} & t = t_1 \\ 0.3e^{i\phi} & t = t_2 \\ 0 & \forall t \end{cases} \qquad abs(FFT) = \begin{cases} \cos(\phi) & t = t_1 \\ 0.3\cos(\phi) & t = t_2 \\ 0 & \forall t \end{cases}$$




le module de signal obtenue par **IFFT** est un spectre de puissance de deux pic, est une signal **périodique**, a partir de ce signal périodique (module de **IFFT**) on peut retourner le spectre de puissance.

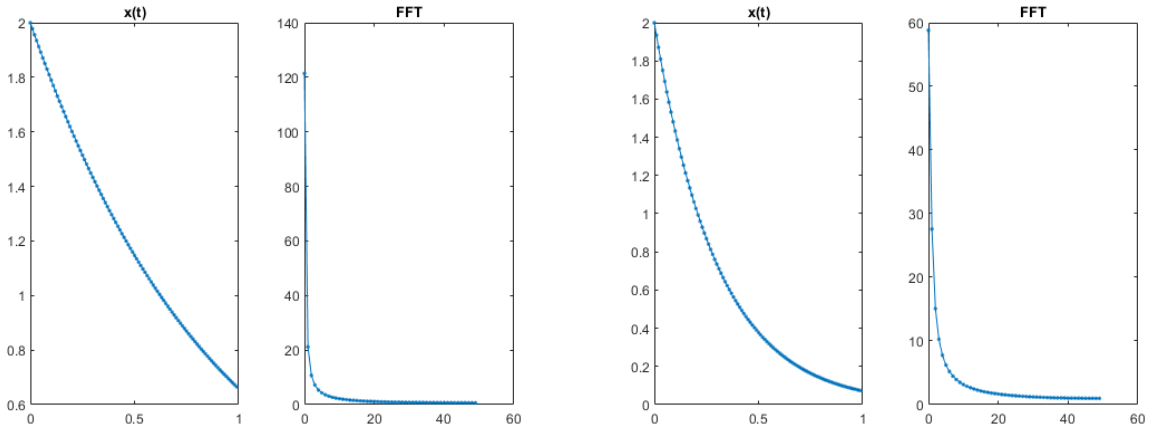
**L'amplitude** de pic de deux spectre de puissance est le même, et **l'amplitude** de signal périodique augment de même façons que le spectre de puissance.

- La phase n'affecte sur le module de FFT au FFT retourner.
- Si le spectre de puissance est plate. sa IFFT est un pic de delta.




En changeant l'emplacement du pic, il semble que la carte de phase soit linéaire avec une pente qui diminue linéairement avec l'emplacement du pic.

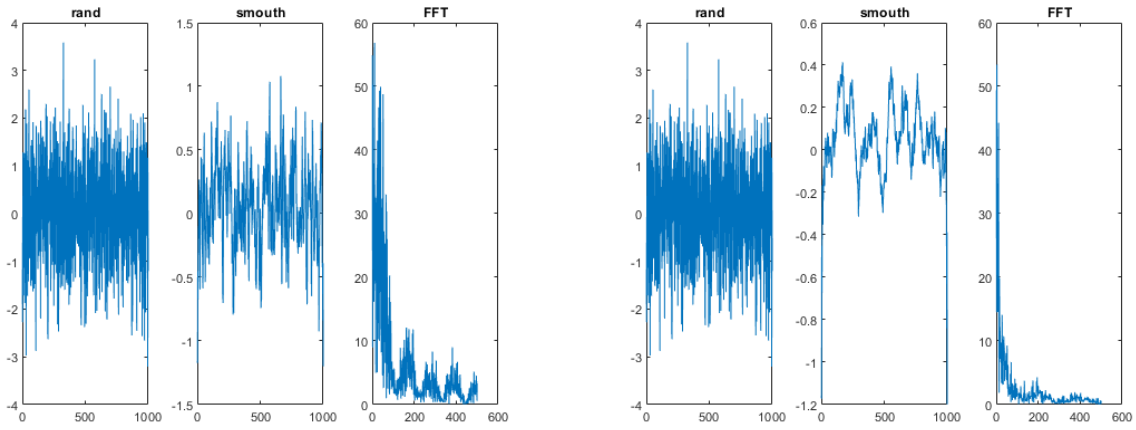
**Fonction exponentiel**




- Si le signal d'origine est large, plus sa transformée de Fourier est étroite.

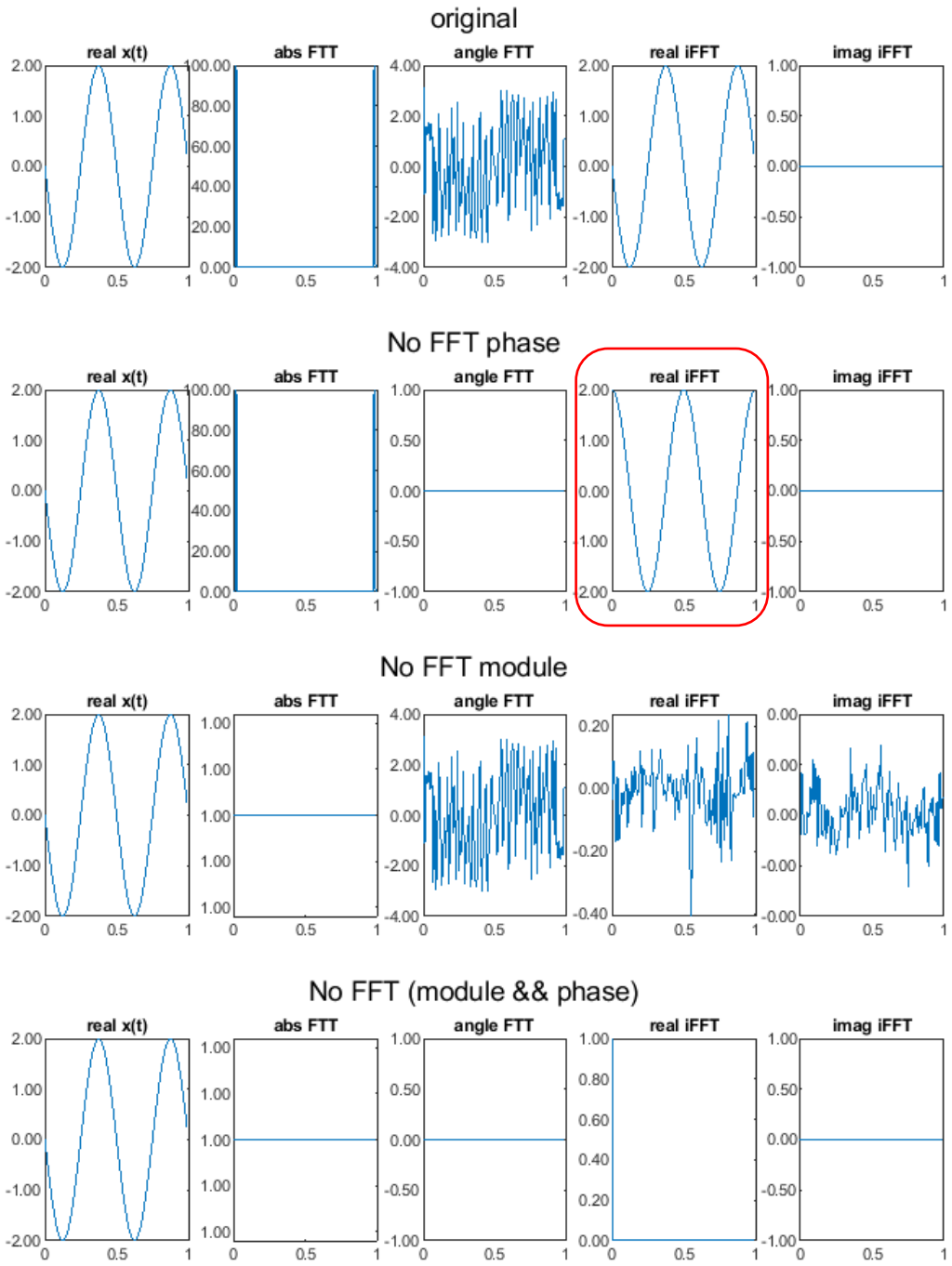
C'est ce qu'on appelle le principe d'incertitude. On  pout localiser le signal ou sa transformée de Fourier, mais pas les deux.

## Bruit+smooth



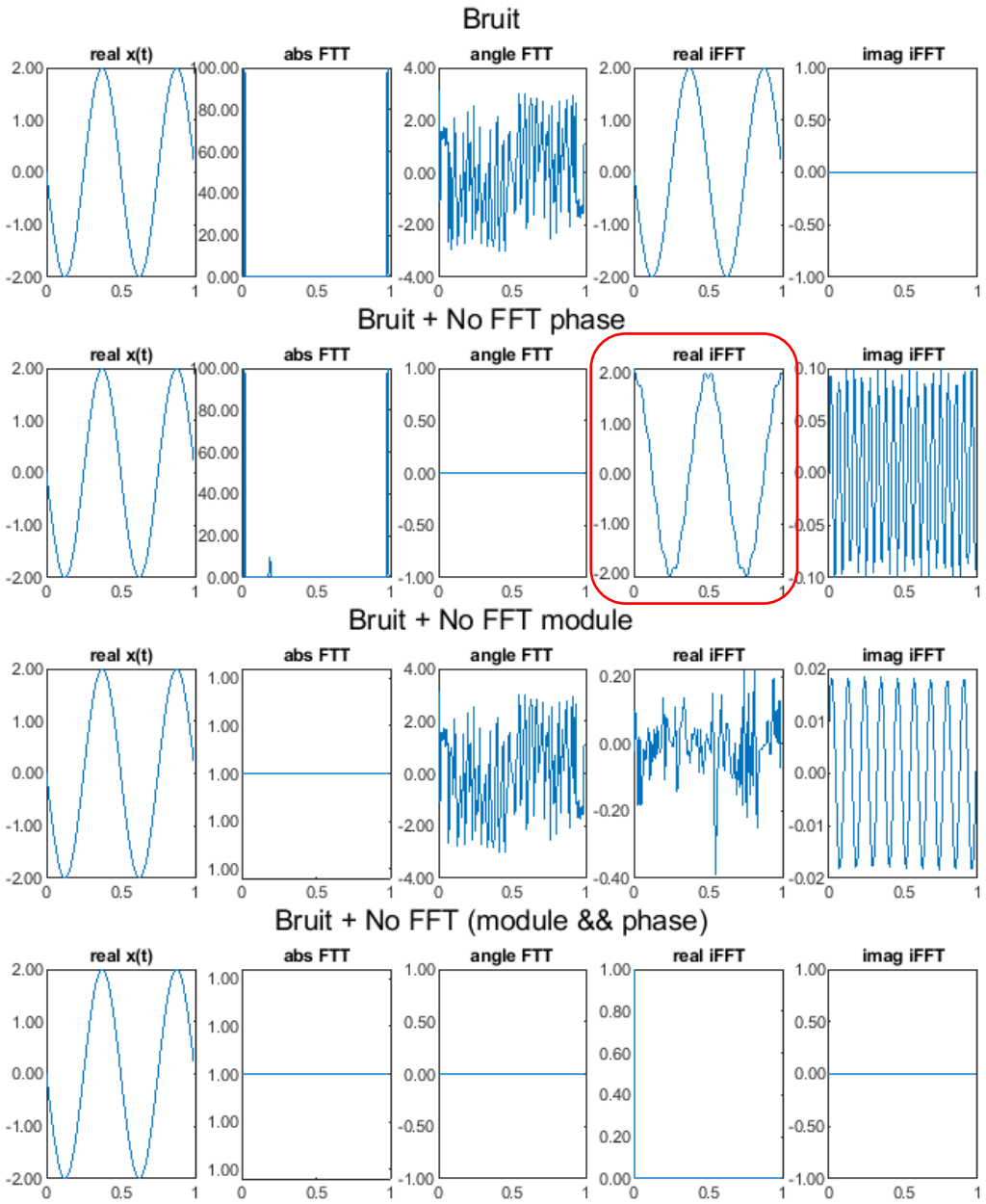

Plus de lissage fait passer le spectre de puissance d'un niveau plat à un pic à fréquence nulle. C'est ce qu'on appelle le bruit « coloré »

## Bruit : supprime phase ou magnitude




Si on supprimé la phase, le signal reconstruit ressemble au signal original mais décalé.

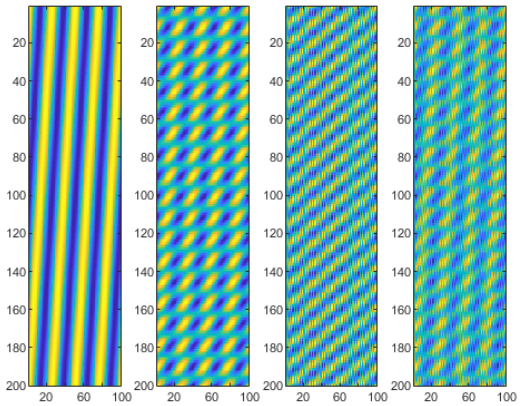
## Ajout de  Bruit



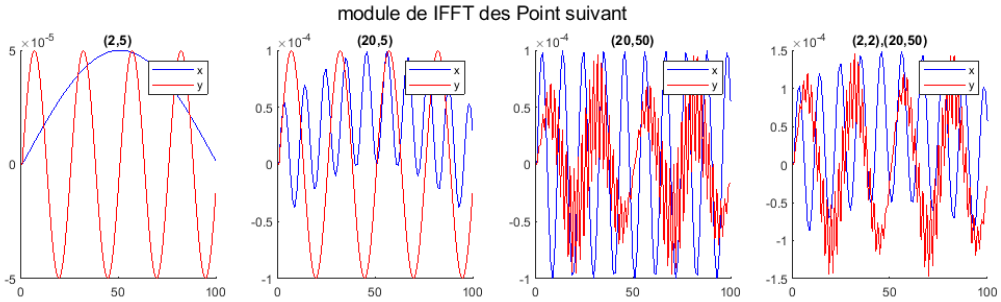

Si on ajoute une **bruit**, le signal reconstruit **ressemble au signal original** mais le pics de magnitude obtenue peuvent introduire des **effets assez désagréables.**

## ❖ FFT2

a) une point de faible x et y égale 1
b) une point de grande x et de faible y égale 1
c) une point de grand x et y égale 1
d) deux point : point faible x et y et une point de grande x et y



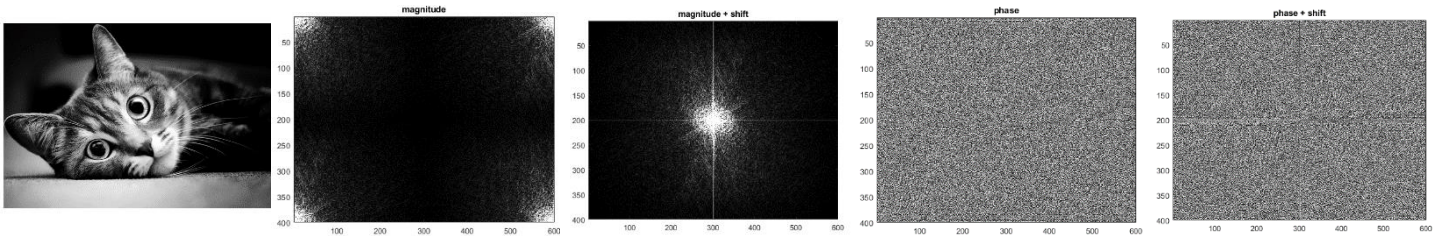
L'augmentation de position de la point choisie (fréquence) est donne une image plus périodique, (la périodicité de x et de y il s'ajout)




suivant une seul axe de image ***iFFT2*** il se transforme sous forme d'une fonction périodique de fréquence élevé, c-t-d plusieurs répétitions successives d'un couleur suivant cette axe.

- ❖ Si on a deux point ou plus, la répétition s'ajoute
- ❖ La phase se déplace l'image restaurer par la fonction ***iFFT2***.

**Quantile**

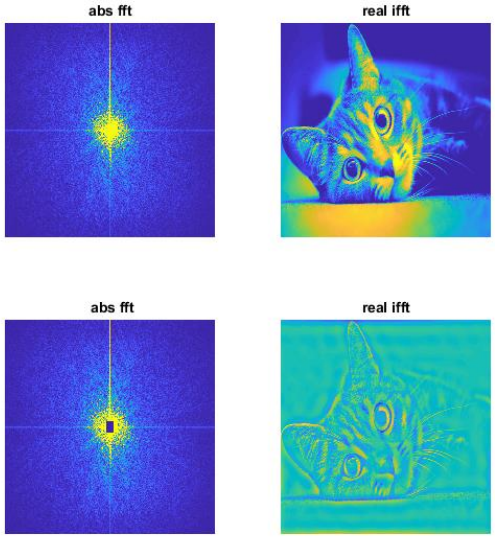



$$\begin{matrix} 4 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 \end{matrix} \quad \textbf{FFTSHIFT} \quad \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

**FFTSHIFT** centre l'image par une déplacement suivant x et y

**QUANTILE** est un option utilise pour améliorer le contraste, dans la fonction *imagesc*(), mais *imshow*() ne peut pas améliorer le contraste, donc il affiche l'image original.

# Box




Les éléments clés à remarquer à partir de la **FFT2** de l'image du chat sont que :

- la magnitude est principalement centrée autour de zéro, c'est-à-dire beaucoup de basses fréquences.
- l'image de phase est difficile à interpréter.
- Les information de l'image sont concentre au centre de l'image fréquentiel

# RESTAURE *img*

Le tableaux $2D$ de $img$ obtenue par $fft2$, est un tableaux des nombre complexe

$$im\ (real) \xrightarrow{fft2} \begin{cases} Z = z_0 e^{i\,\phi} \\ Z = a + ib \end{cases} \xrightarrow{ifft2} im'(complex)$$

A partir de ce nombre complexe $Z$ on tester est ce qu'on peut restaurer l'image grâce au module seulement $z_0$ ou facteur d'amplitude $e^{i\,\phi}$ ou la phase $\phi$.

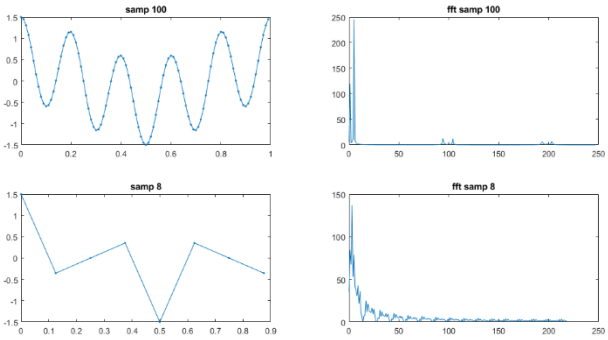
$ifft2$ donne aussi un tableaux des nombres complexe, on peur l'afficher le module de cette image(tableaux) obtenu par $ifft2$ ou on affiche la phase.

| | Image de module | Image de phase |
|---|---|---|
| $z_1 = ifft2(z_0 e^{i\,\phi})$ | 100% | 1% |
| $z_2 = ifft2(z_0)$ | 0% | 0% |
| $z_3 = ifft2(\phi)$ | 50% | 0% |
| $z_4 = ifft2(e^{i\,\phi})$ | 20% | 1% |

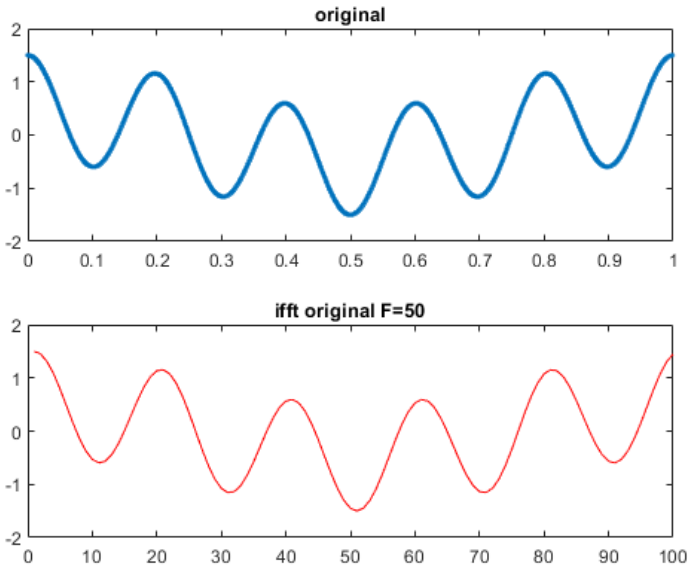
- La magnitude nous dit ce qui se passe (en termes de fréquences).
- La phase nous dit "où" ça se passe.

Ainsi, alors que la magnitude était suffisante pour des signaux simples comme un cosinus, elle échoue pour des signaux complexes comme des images naturelles, car alors il importe vraiment où les oscillations sont localisées.

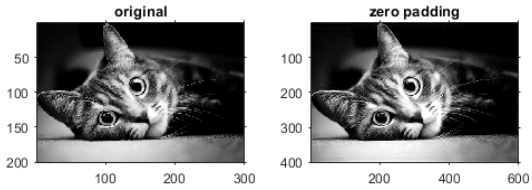
# ÉCHANTILLONNAGE ET REPLIEMENT




les fréquences augment si on déminer le nombre de l'échantillonnage

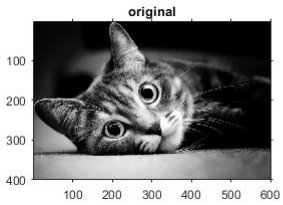


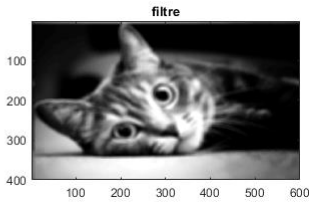

Le signal est ressemblé que le signal original mais avec un nouveau fréquence.




Le zéro padding signifie qu'ils ajoutent des zéros aux côtés de la FFT avant de reconstruire le signal. Cela permet de reconstruire l'image avec une plus grande résolution que ce qui était possible dans le domaine fréquentiel et est lié au concept d'interpolation.

Puisque la convolution dans l'espace réel est la même que la multiplication dans l'espace de Fourier, est la multiplication est plus facile que la convolution, pour faire un filtre linéaire simple.




Création de filtre utilisant :
Multiplication de
$Z = ifft2(0.1 \cdot I \quad \cdot im)$

Pour calculer la projection d'une surface sur un axe, nous mesurons la somme de toutes les sections de la surface qui sont parallèles à l'axe.

$$\begin{matrix}1&0&0&0&0&1\\0&1&0&0&0&0\\0&0&1&0&1&0\\1&0&0&0&0&1\end{matrix}$$

$$\begin{cases} projection\ sur\ un\ axe\ horizental \\ projection\ ur\ un\ axe\ vertical \\ projection\ sur\ axe\ y = 1x \end{cases} \begin{matrix}2&0&0&0&0&1\\2&0&0&0&0&1\\2&0&0&0&0&1\end{matrix}$$

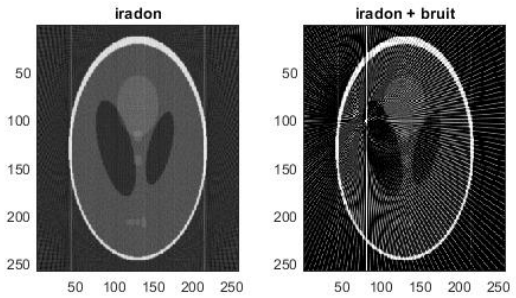
- projection sur un axe horizontal (axe y dans Matlab) est un vecteur $2 \quad 1 \quad 2 \quad 2$
- projection sur un x c'est $2 \quad 1 \quad 1 \quad 0 \quad 1 \quad 2$
- projection sur un axe $f(x) = x$, c'est: $1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$

est impossible de reconstruire une image par une seul projection, il faut tester tous les angles.

# TDM

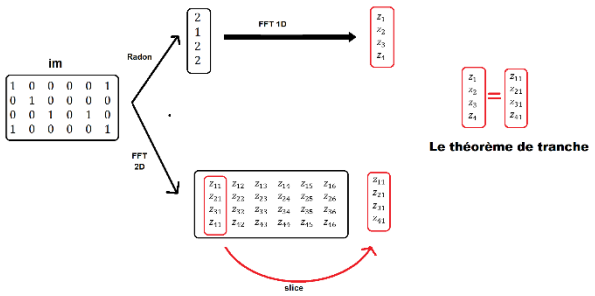
Ces codes sont des exemples d'opérations dans le domaine de la reconstruction d'images en imagerie médicale, en utilisant la transformée de Radon et la transformée d'iradon.

Le premier code utilise la transformée de Radon et d'iradon pour simuler le processus de mesure et la reconstruction d'un objet à partir de mesures. Le code définit un objet appelé "im", qui est un phantom 2D (une image test standard utilisée pour les tests d'imagerie). Ensuite, une simulation d'implant métallique est ajoutée à "im" en définissant un pixel spécifique à une valeur de 100. La transformée de Radon est alors appliquée à "im", ce qui correspond à la projection de l'objet sur des lignes à différents angles. Enfin, la transformée d'iradon est utilisée pour reconstruire l'objet à partir des mesures. Le résultat montre des artefacts frappants causés par l'implant métallique simulé.




$im(100,80) = 100$; ça me donne un point d'intensité plus grand que les autres point au moins 100 fois, ce qui translate un artéfact

Le deuxième code montre comment la transformée de Radon est liée à la transformée de Fourier via le théorème de tranche de projection. Le code utilise l'objet "im" défini dans le premier code et prend une projection de celui-ci sur l'axe x (le long de l'axe y). Ensuite, une transformée de Fourier 1D et 2D est effectuée sur la projection et sur "im" respectivement. Enfin, une tranche à travers la transformée de Fourier 2D est prise pour montrer comment la projection est liée à la transformée de Fourier.

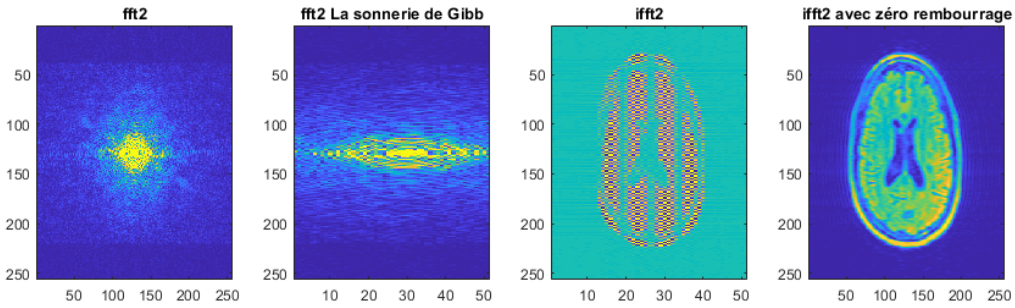



On peut utiliser la transformée de Fourier pour faire une transformation inverse du radon d'une image.

# IRM

## LA SONNERIE DE GIBB

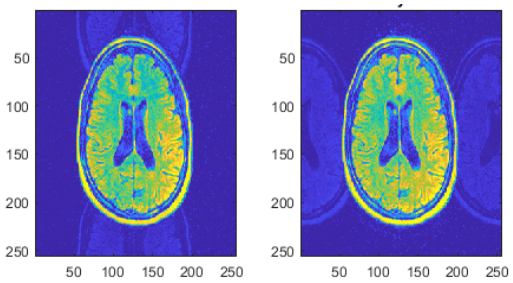
Le troisième code utilise une image IRM d'un cerveau pour montrer l'effet de la sonnerie de Gibb.




Tout d'abord, nous chargeons une image appelée "cerveau". Ensuite, nous mesurons la transformée de Fourier 2D de l'objet et la stockons dans une variable appelée "data". Ensuite, nous simulons la sonnerie de Gibbs en coupant une partie de l'espace k le long d'un des axes. Cela signifie que nous n'avons pas assez de temps pour acquérir toutes les données hautes fréquences. Enfin, nous reconstruisons l'objet complet en utilisant l'inverse de la transformée de Fourier et affichons l'image. Vous devriez pouvoir voir les artefacts de Gibbs ringing le long de l'axe horizontal.

## FANTOME

Le quatrième code montre l'effet d'une image fantôme. La transformée de Fourier 2D est effectuée sur l'image IRM d'un cerveau, puis une ligne sur deux dans l'espace k est modulée en intensité pour simuler une erreur de mesure. La reconstruction de l'objet est effectuée à l'aide de la transformée d'iradon et montre une duplication de l'image.Le code montre également l'effet de la modulation d'intensité sur les deux dimensions de l'image.

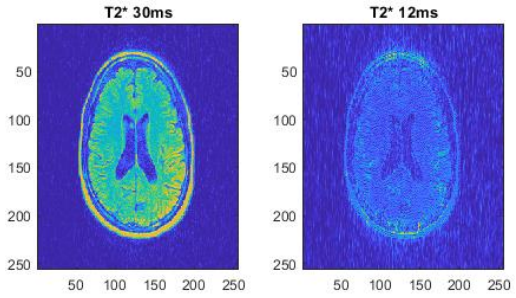


Les images fantômes se produisent dans l'analyse d'images en utilisant la transformée de Fourier. Cela se produit lorsque certaines lignes de l'espace k sont modulées en intensité différemment, ce qui se traduit par une duplication de l'image originale avec une répétition à des intervalles de N/2 pixels. Cela se produit car la modulation de certaines lignes dans l'espace k est similaire à ajouter une image qui est échantillonnée à des intervalles de 2 pixels, ce qui se traduit par une répétition de l'image dans l'espace de l'image originale.

## T2* FLOU

Ce code a pour objectif de simuler un artefact appelé flou $T2*$ en **imagerie par résonance magnétique (IRM).**

T2* est une constante de temps qui représente le taux de décroissance du signal mesuré dans l'espace k (un espace de représentation de l'image). Le signal dans l'espace k est multiplié par une décroissance exponentielle pour simuler le flou T2*, ce qui se traduit par une convolution de l'image avec la transformée de Fourier de l'exponentielle (fonction lorentzienne). En fin de compte, cela produit une image brouillée.




Cette artefact se produit lorsque nous acquérons des données pour faire une image IRM. Pendant ce processus, le signal qui nous sert de mesure décroît au fil du temps, ce qui entraîne un flou dans l'image. Pour simuler cet effet, nous faisons une opération mathématique appelée **convolution**, qui consiste à brouiller l'image en la combinant avec une autre fonction mathématique appelée fonction lorentzienne. Plus la constante de temps **T2* est courte**, plus l'image **sera floue**.

# OPERATIONS APPLIQUEES A "SHEPP-LOGAN PHANTOM"

Il existe plusieurs opérations qui peuvent être appliquées à l'objet "Shepp-Logan phantom" dans Matlab. Certaines des opérations courantes incluent :

- **Transformée de Fourier rapide (FFT)** : pour visualiser la distribution de fréquence dans l'objet.
- **Transformée de Fourier inverse (IFFT)** : pour retourner à la représentation spatiale.
- **Convolution :** pour brouiller ou flouter l'objet.
- **Déconvolution :** pour enlever le flou ajouté par une opération de convolution.
- **Filtrage :** pour supprimer les artefacts ou les bruits dans l'objet.
- **Interpolation :** pour augmenter la résolution spatiale de l'objet.
- **Normalisation :** pour normaliser l'intensité du signal pour un meilleur affichage.

Voici un exemple des opérations d'image avec Matlab utilise ce fantôme :

## Filtrage :

```
im = phantom(256); % image fictive appelée "fantôme Shepp-Logan"

% Affichage de l'image avant le filtrage
figure, imagesc(im), colormap gray;
title('Image originale');

% Définir un filtre de passe-bas
filter = ones(5, 5);
filter = filter / sum(filter(:)); % normalisation pour conserver l'intensité totale

% Appliquer le filtre en utilisant la convolution
im_filtered = conv2(im, filter, 'same');

% Affichage de l'image filtrée
figure, imagesc(im_filtered), colormap gray;
title('Image filtrée');
```

## Interpolation:

```matlab
% Initialisation de l'image
im = fantome;

% Taille de l'image originale
[rows, cols] = size(im);

% Facteur d'interpolation
interp_factor = 2;

% Calcul des nouvelles dimensions de l'image interpollée
new_rows = interp_factor * rows;
new_cols = interp_factor * cols;

% Matrices pour les coordonnées de la nouvelle grille d'interpolation
[X, Y] = meshgrid(1:1/interp_factor:cols, 1:1/interp_factor:rows);

% Interpolation de l'image
interp_im = interp2(im, X, Y, 'cubic');

% Affichage de l'image interpollée
figure, imshow(interp_im, [])
title('Image interpollée')
```

## Normalisation:

```matlab
im = fantome;
normalizedIm = mat2gray(im); % Normalize the image between 0 and 1
% Alternatively, you can use the following line
% to normalize the image between a specified range:
normalizedIm = imadjust(im, [min(im(:)) max(im(:))], [0 1]);
imshow(normalizedIm)
```