

IMN-359

Méthodes mathématiques du traitement d'images

-

Conclusion

IMN-359

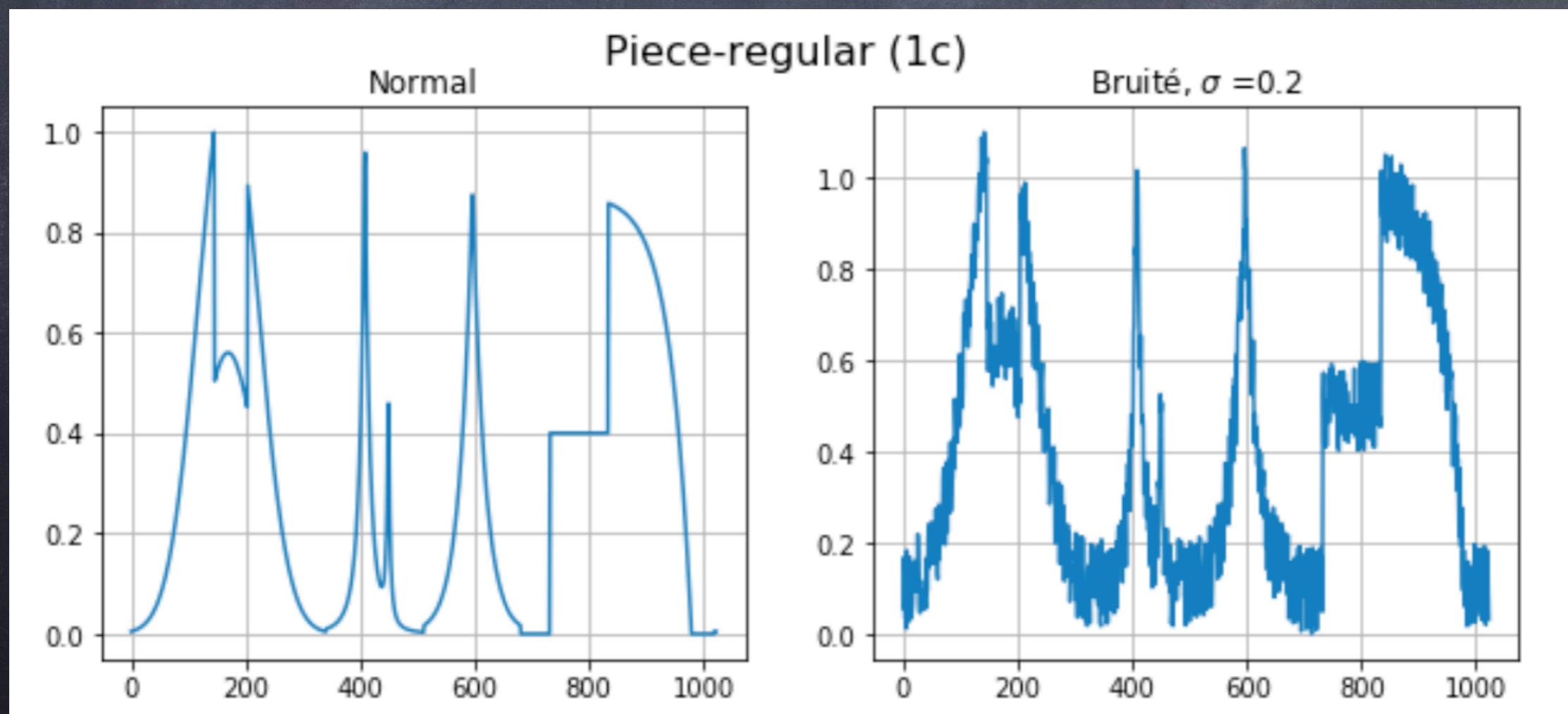
- ⦿ Cours de math (théorèmes, preuves, questions théoriques sur le TPs et les examens)
- ⦿ Cours d'informatique (programmation Python, algorithmique, analyse de complexité)
- ⦿ Cours d'imagerie (applications sur des signaux réels)
 - ⦿ signal 1D
 - ⦿ images 2D

IMN-359

- ➊ Si vous faites vos TPs, les comprenez, vous aurez un A et plus assuré
- ➋ Retour sur TP3 et TP4 crucial pour l'examen final

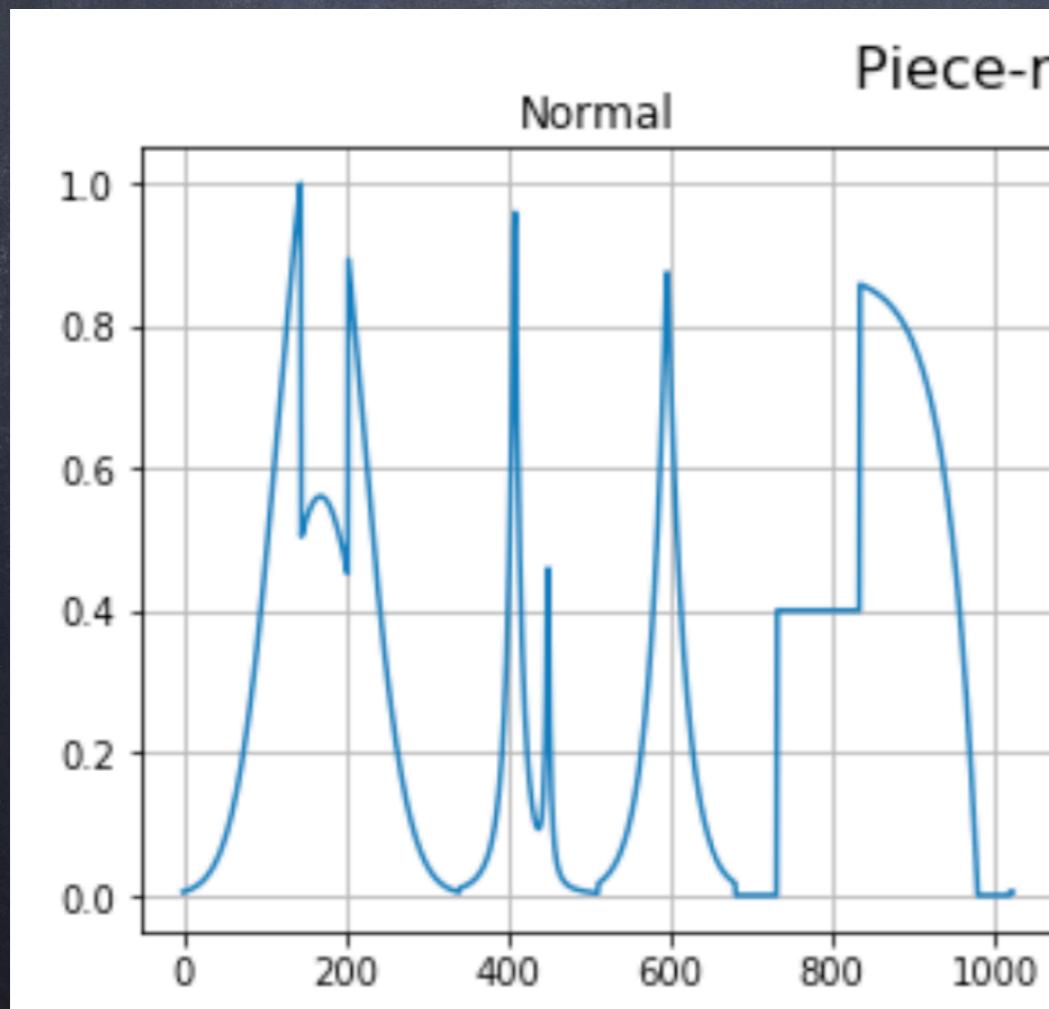
Retour sur le TP3

- On peut bruité un signal en ajoutant un petit random sur le signal

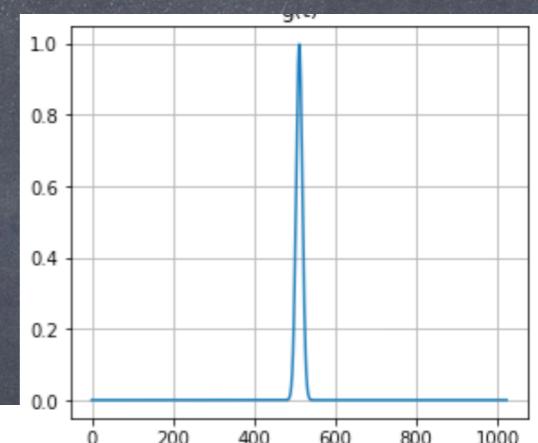
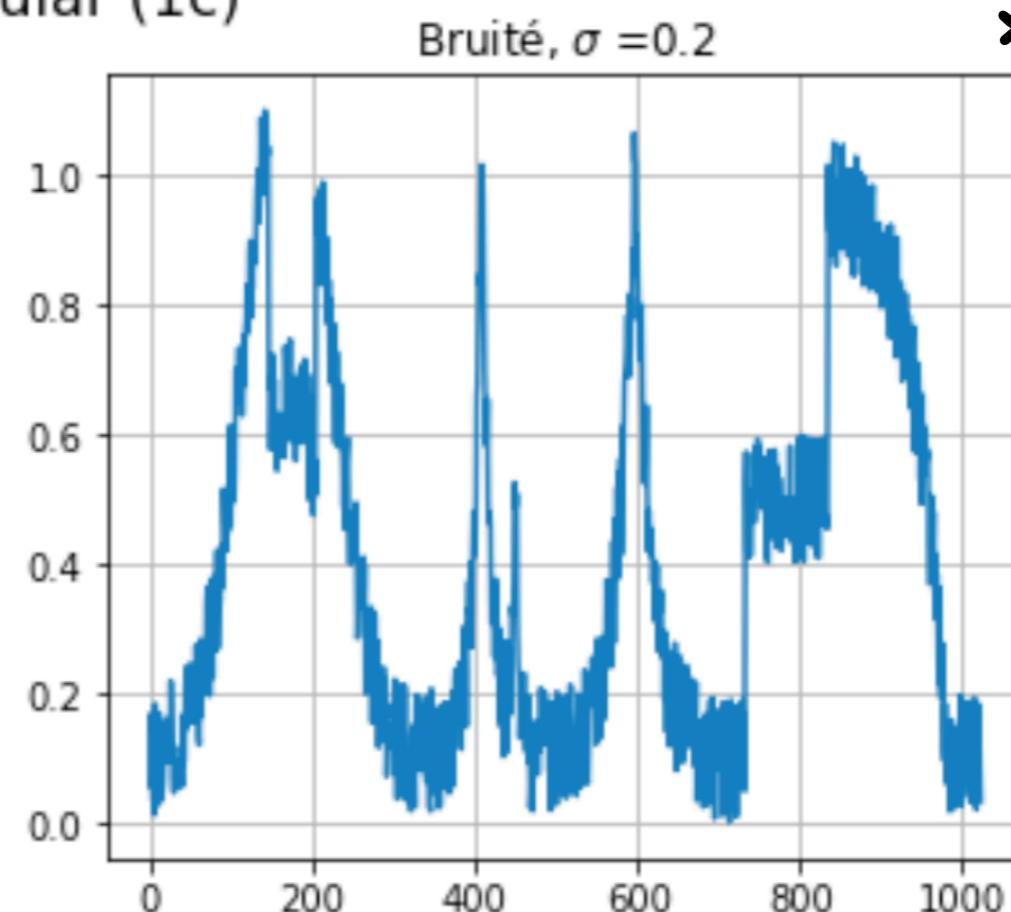


Retour sur le TP3

- On peut bruité un signal en ajoutant un petit random sur le signal
- En convoluant avec une Gaussienne on peut débruité le signal.
`convolve(Gaussienne, piece-regular)`



Piece-regular (1c)



- En convoluant avec une Gaussienne on peut débruité le signal.

`convolve(Gaussienne, piece-regular)`

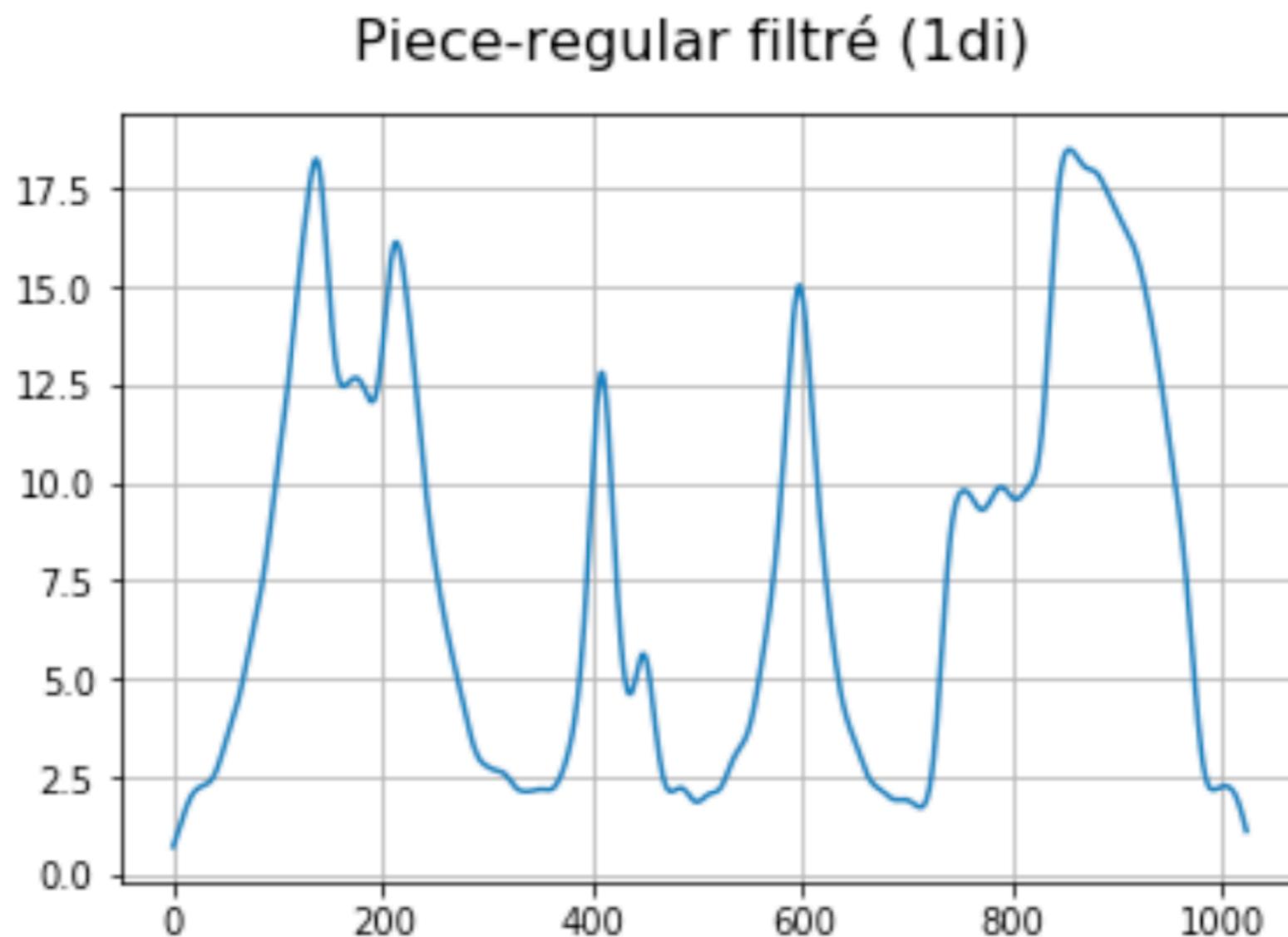
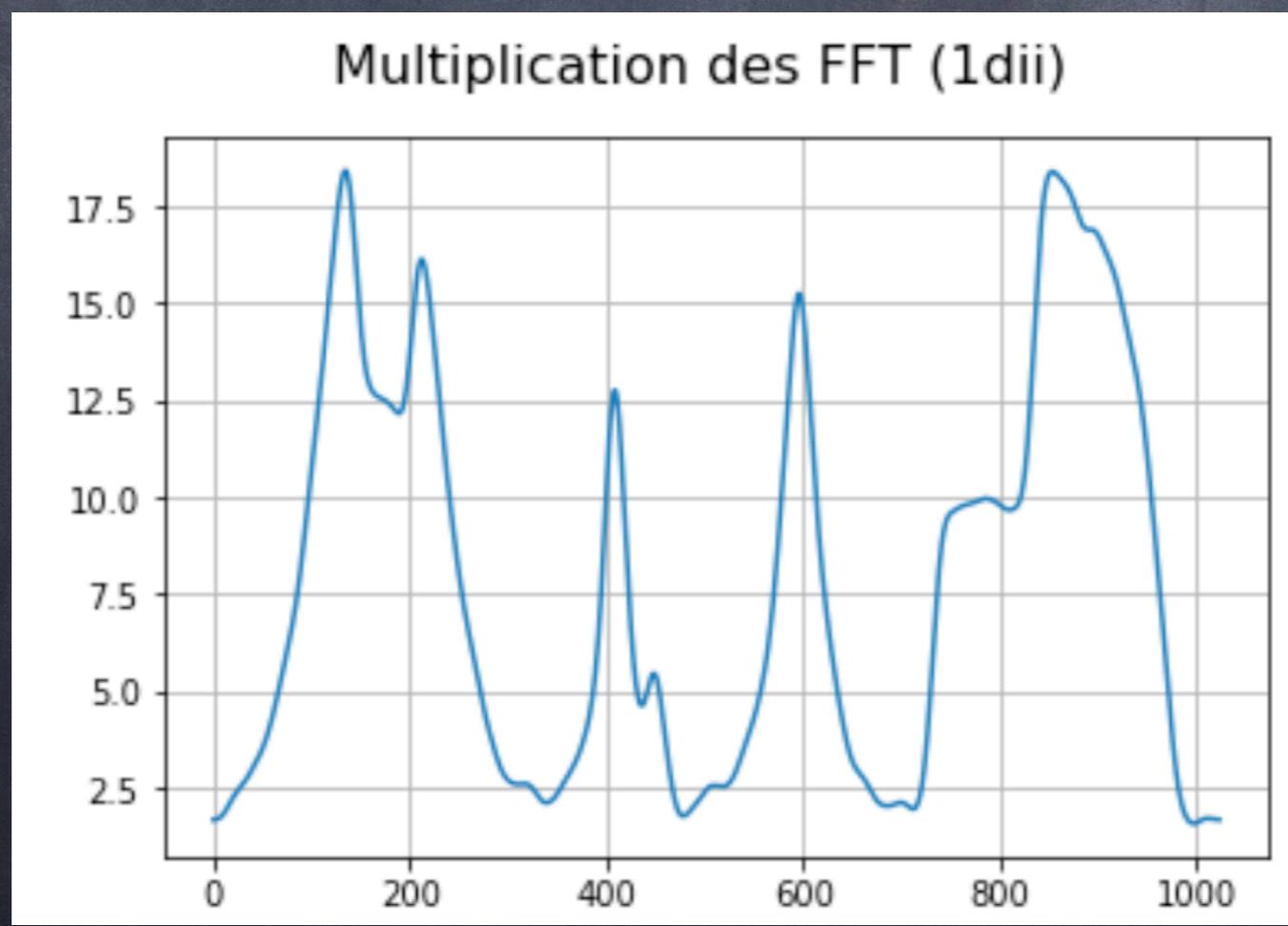


FIGURE 4 – Convolution du signal 'piece-regular' bruité et du filtre Gaussien

Retour sur le TP3

- Le théorème de la convolution dit qu'une convolution dans le monde temporel est équivalent à une multiplication dans le monde de Fourier.
- `ifft(fft(Gaussienne) . fft(piece-regular)) = convolve(Gaussienne, piece-regular)`



- ➊ Donc, notre convolution qui débruite le piece-regular est, en fait, un filtrage passe-bas dans le monde des fréquences

- ➊ Donc, notre convolution qui débruite le piece-regular est, en fait, un filtrage passe-bas dans le monde des fréquences
- ➋ La TF de la Gaussienne est une Gaussienne qui multiplie les fréquences de la TF du piece-regular et qui laisse passer que les basses fréquences

Thm de la convolution

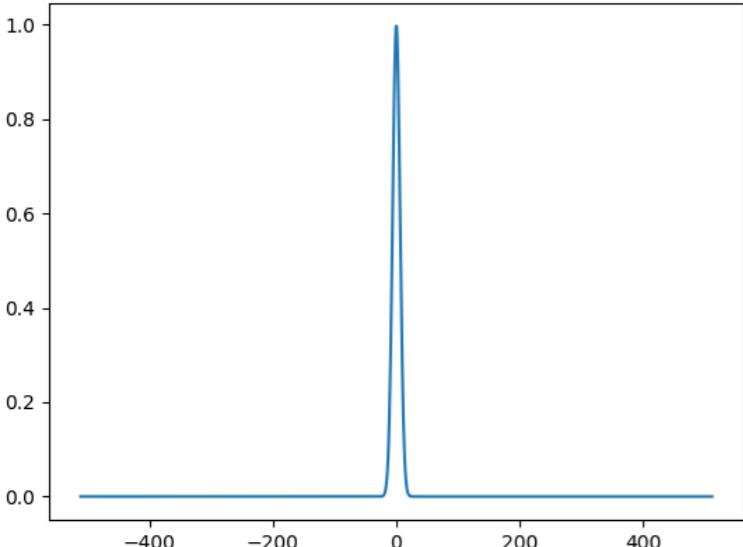
Thm de la convolution

Temps

Fréquences

Temps

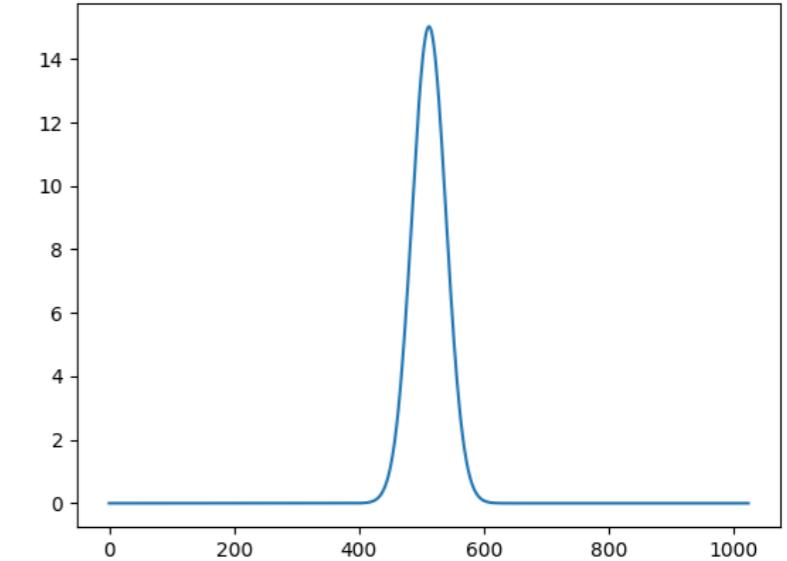
Gaussienne



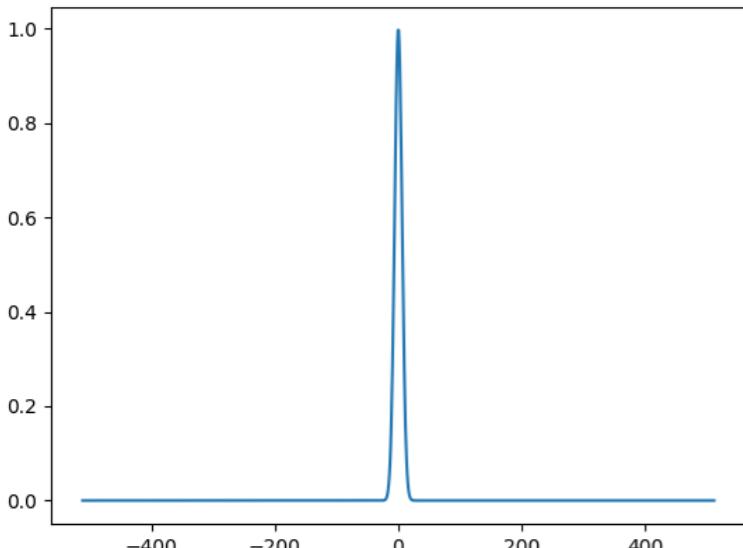
Thm de la convolution

Fréquences

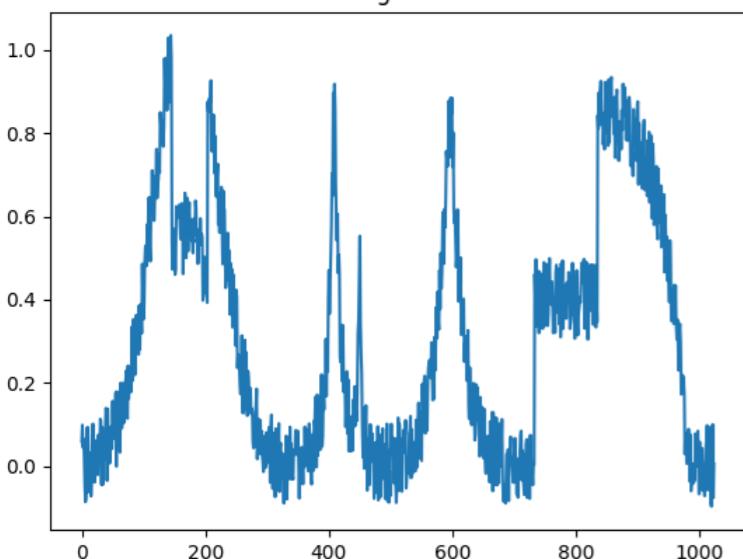
fftshift(np.abs(fft(Gaussienne)))



Gaussienne



Piece-regular bruité

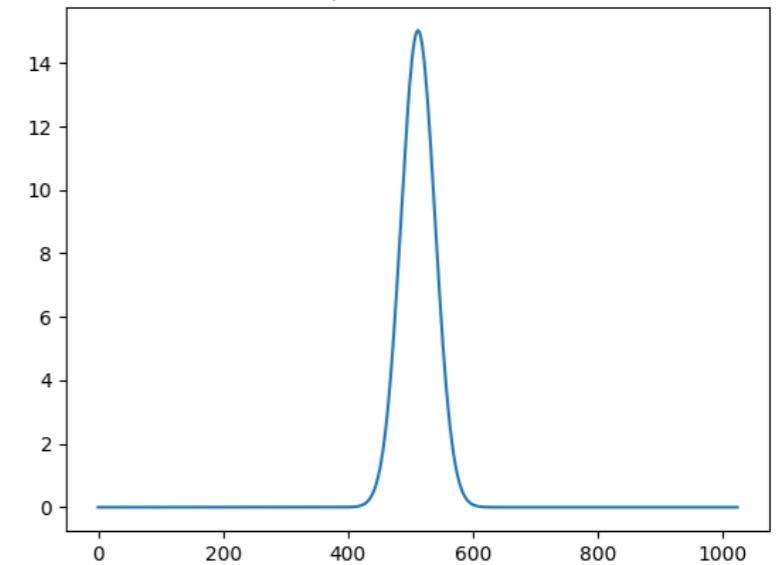


Thm de la convolution

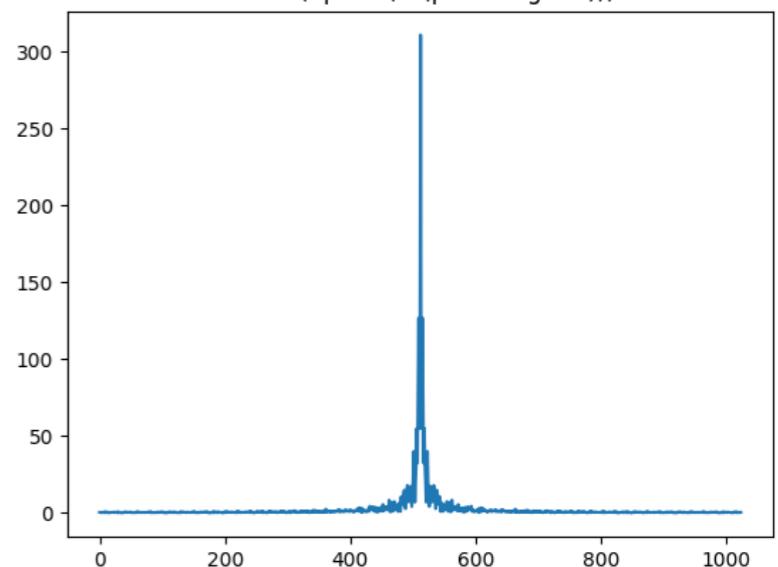
Temps

Fréquences

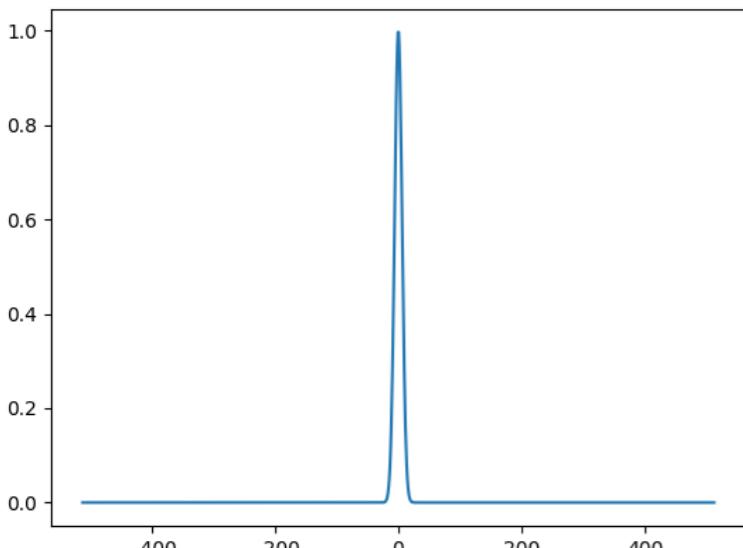
fftshift(np.abs(fft(Gaussienne)))



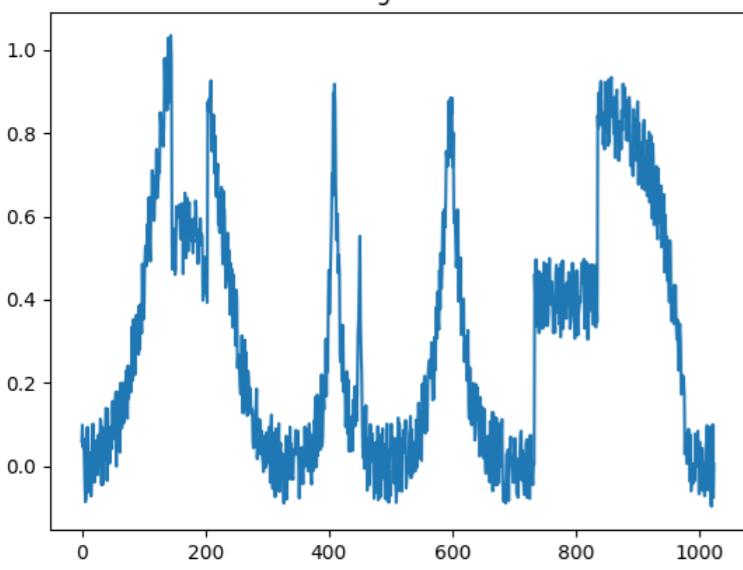
fftshift(np.abs(fft(piece-regular)))



Gaussienne

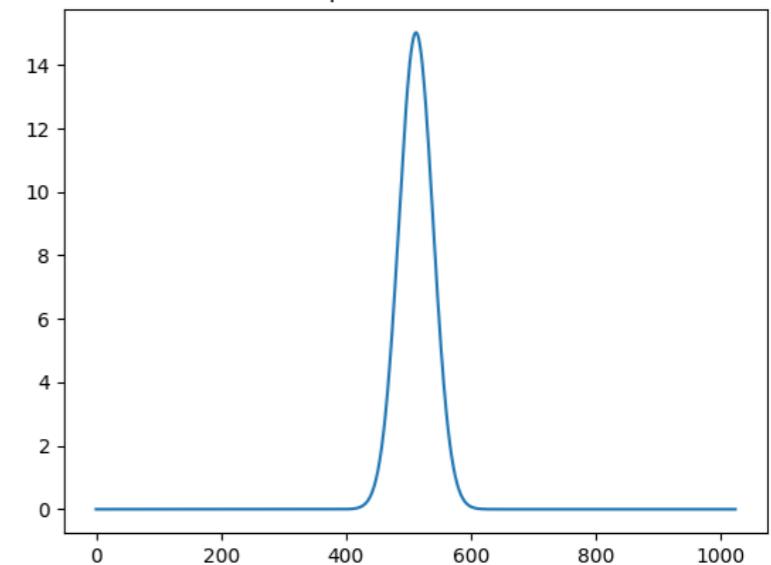


Piece-regular bruité

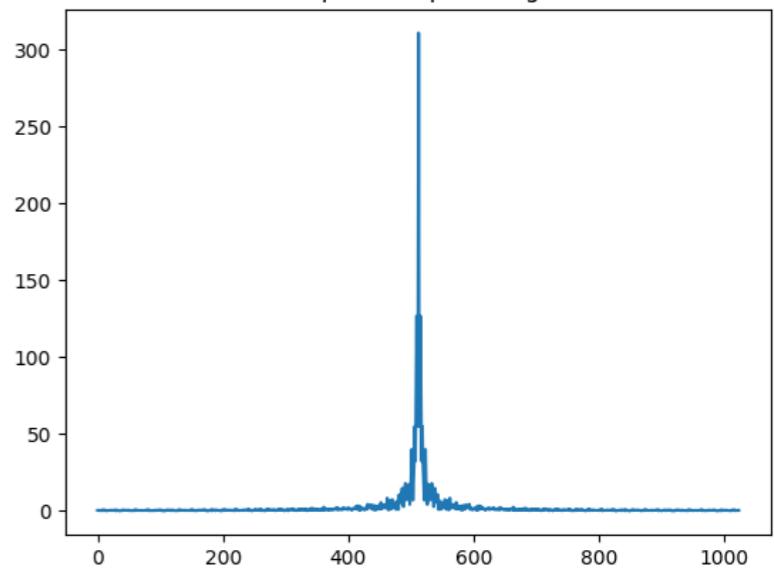


Thm de la convolution

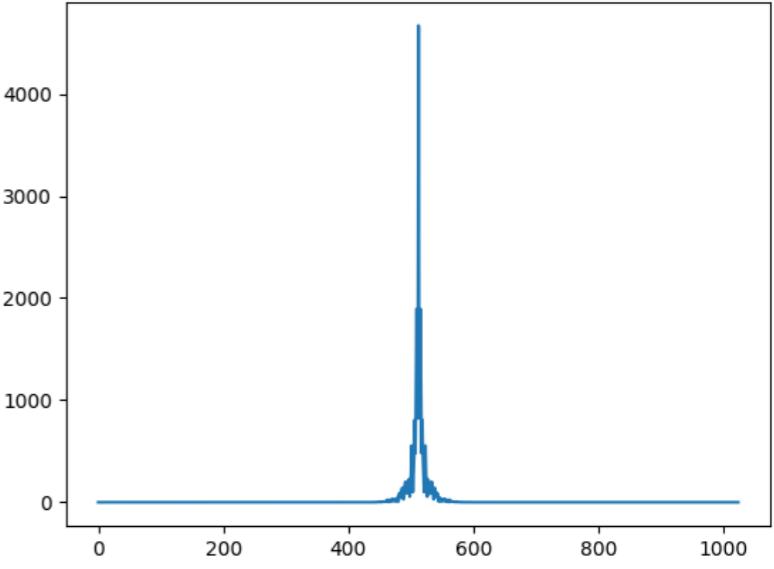
fftshift(np.abs(fft(Gaussienne)))



fftshift(np.abs(fft(piece-regular)))



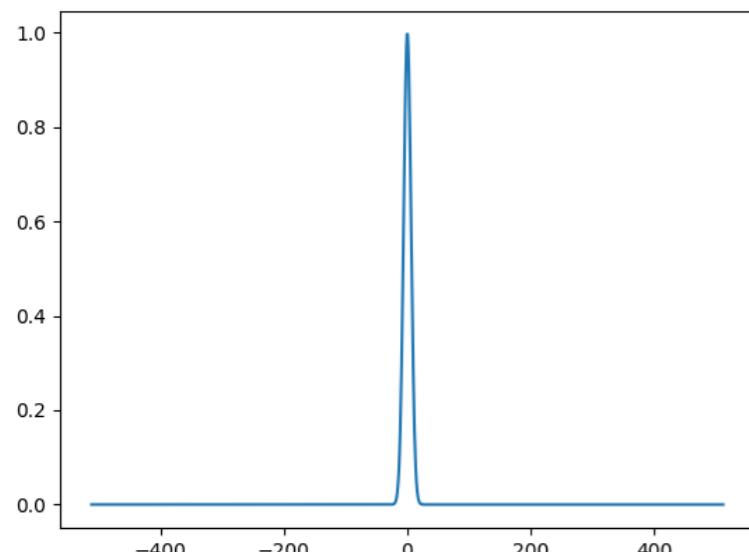
fftshift(np.abs(fft(Gaussienne)*fft(piece-regular)))



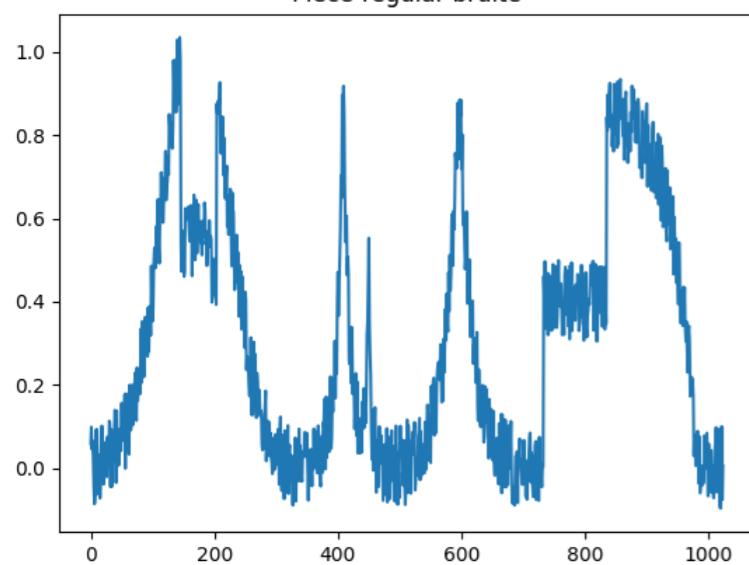
Temps

Fréquences

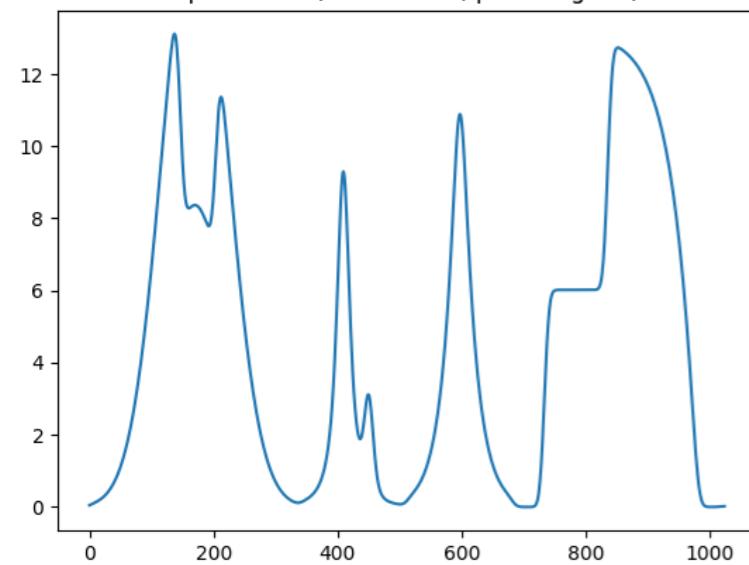
Gaussienne



Piece-regular bruité

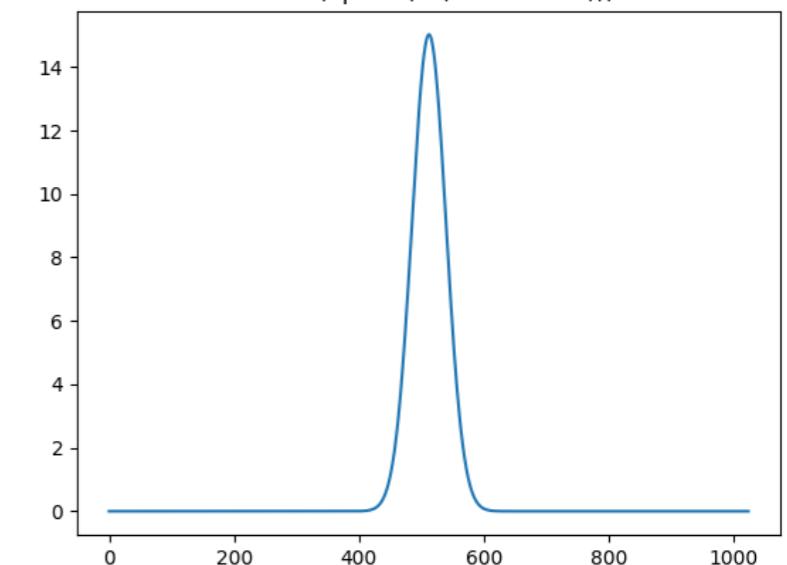


np.convolve(Gaussienne, piece-regular)

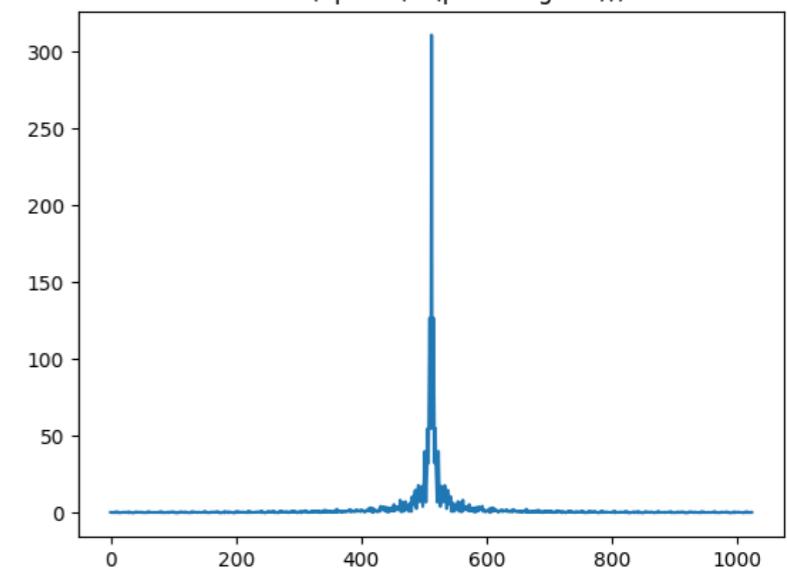


Thm de la convolution

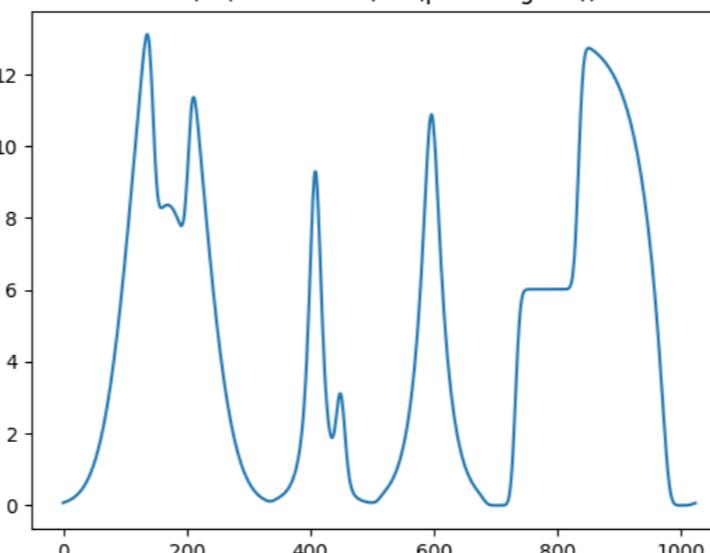
fftshift(np.abs(fft(Gaussienne)))



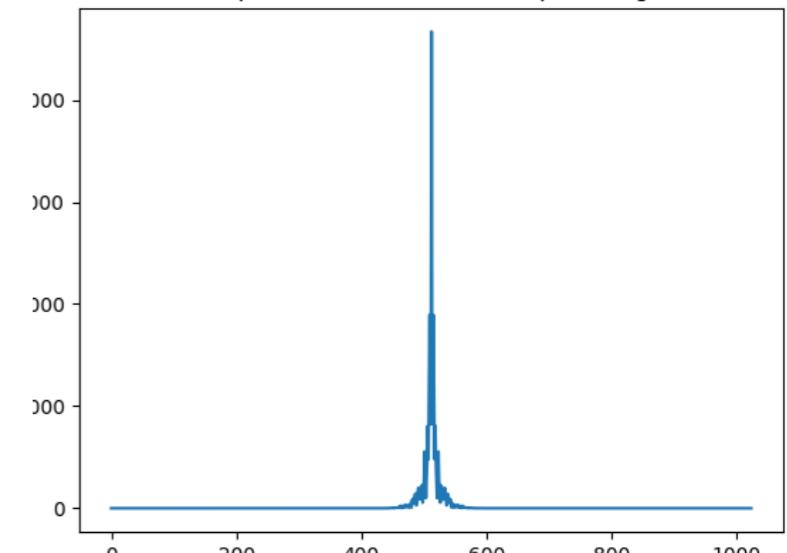
fftshift(np.abs(fft(piece-regular)))



ifft(fft(Gaussienne)*fft(piece-regular))



fftshift(np.abs(fft(Gaussienne)*fft(piece-regular)))



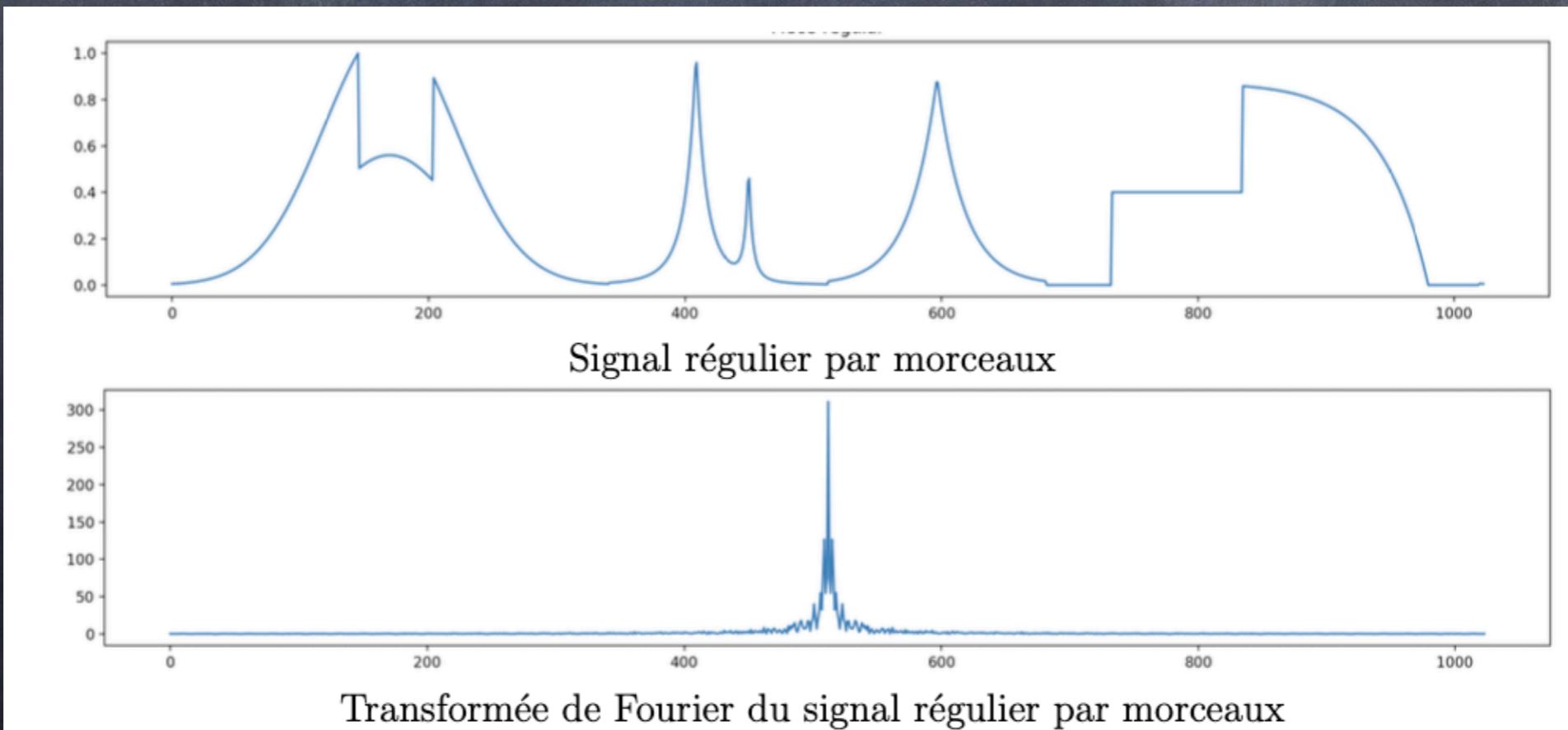
Temps

<= ifft

Fréquences

Retour sur le TP3

- L'énergie est conservée entre le monde temporel et les fréquences. La somme des intensités au carré du piece-regular sont égales à la somme des fréquences au carré divisé pour N (nombre de pts dans le piece-regular)
- $\text{np.sum(piece-regular}^{**2}) = \frac{1}{N} * \text{np.sum(np.abs(fft(piece-regular}))}^{**2}$



IMN-359

- ⦿ Comprenez le théorème d'échantillonnage!
- ⦿ Diapos, demo07
- ⦿ TP3_1a
- ⦿ TP3 question 3

- C'est quoi échantillonné?

- Multiplié par un peigne de Dirac dans l'espace à chaque temps t_s et de largeur X . La TF de ce peigne de Dirac est un peigne de Dirac de largeur $2/t_s$ avec X fréquences espacées à chaque $1/X$ (fftfreq)

• C'est quoi échantillonner?

- Multiplié par un peigne de Dirac dans l'espace à chaque temps t_s et de largeur X . La TF de ce peigne de Dirac est un peigne de Dirac de largeur $2/t_s$ avec X fréquences espacées à chaque $1/X$ (fftfreq)
- Donc, dans Fourier prendre des copies de la TF du signal à chaque bande de fréquences

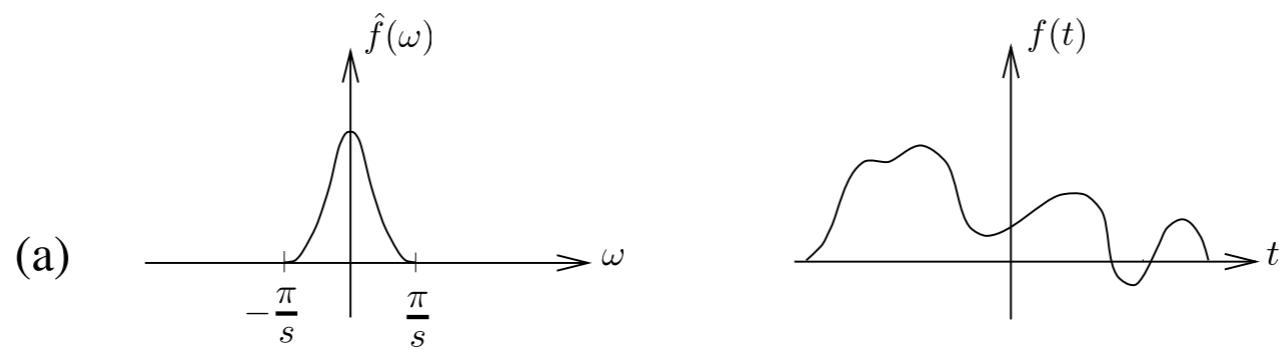
• C'est quoi échantillonné?

- Multiplié par un peigne de Dirac dans l'espace à chaque temps t_s et de largeur X . La TF de ce peigne de Dirac est un peigne de Dirac de largeur $2/t_s$ avec X fréquences espacées à chaque $1/X$ (fftfreq)
- Donc, dans Fourier prendre des copies de la TF du signal à chaque bande de fréquences
- Ensuite, multiplié par une fonction porte sur la bande de fréquences
(ce qui équivaut une convolution dans l'espace par un sinus cardinal – sinc)

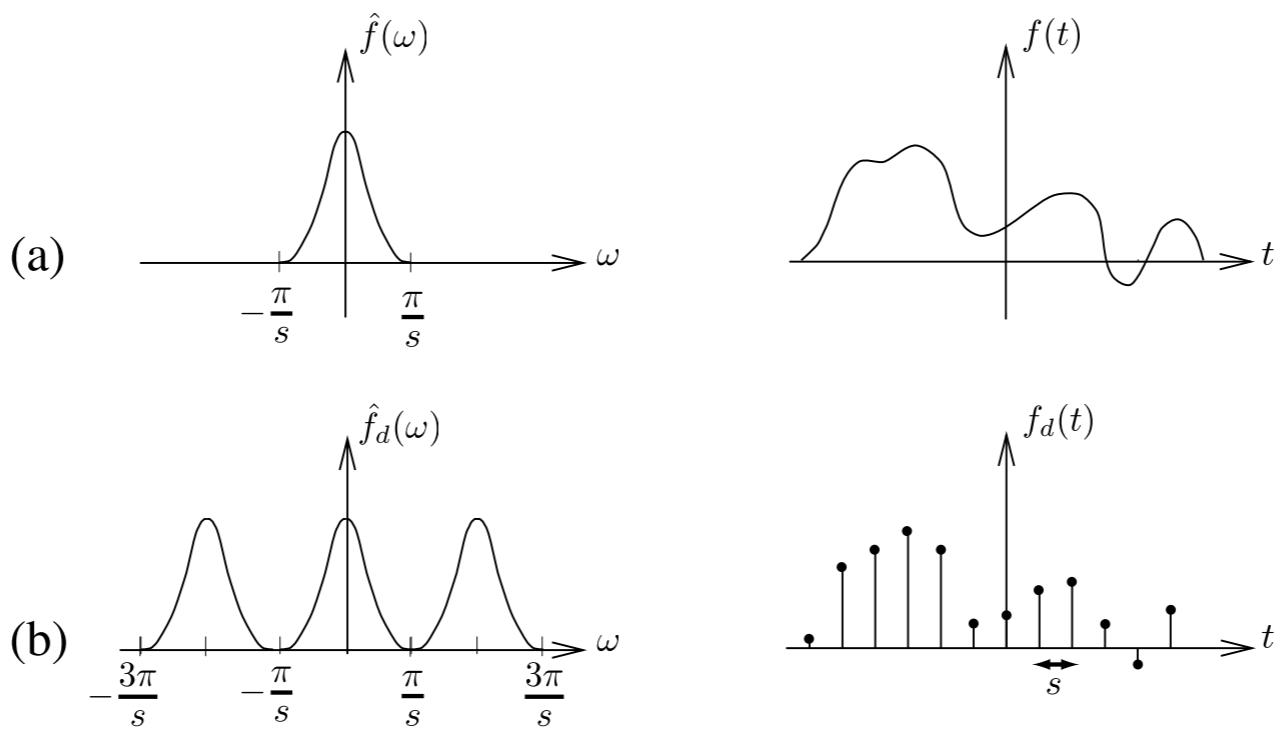
- C'est quoi échantillonné?

- Il faut donc que toutes les vraies fréquences du signal soit contenue dans la bande de fréquences prescrites par la TF du peigne de Dirac (fftfreq)
- => il faut au moins $2^*(\text{fréquence max du signal})$

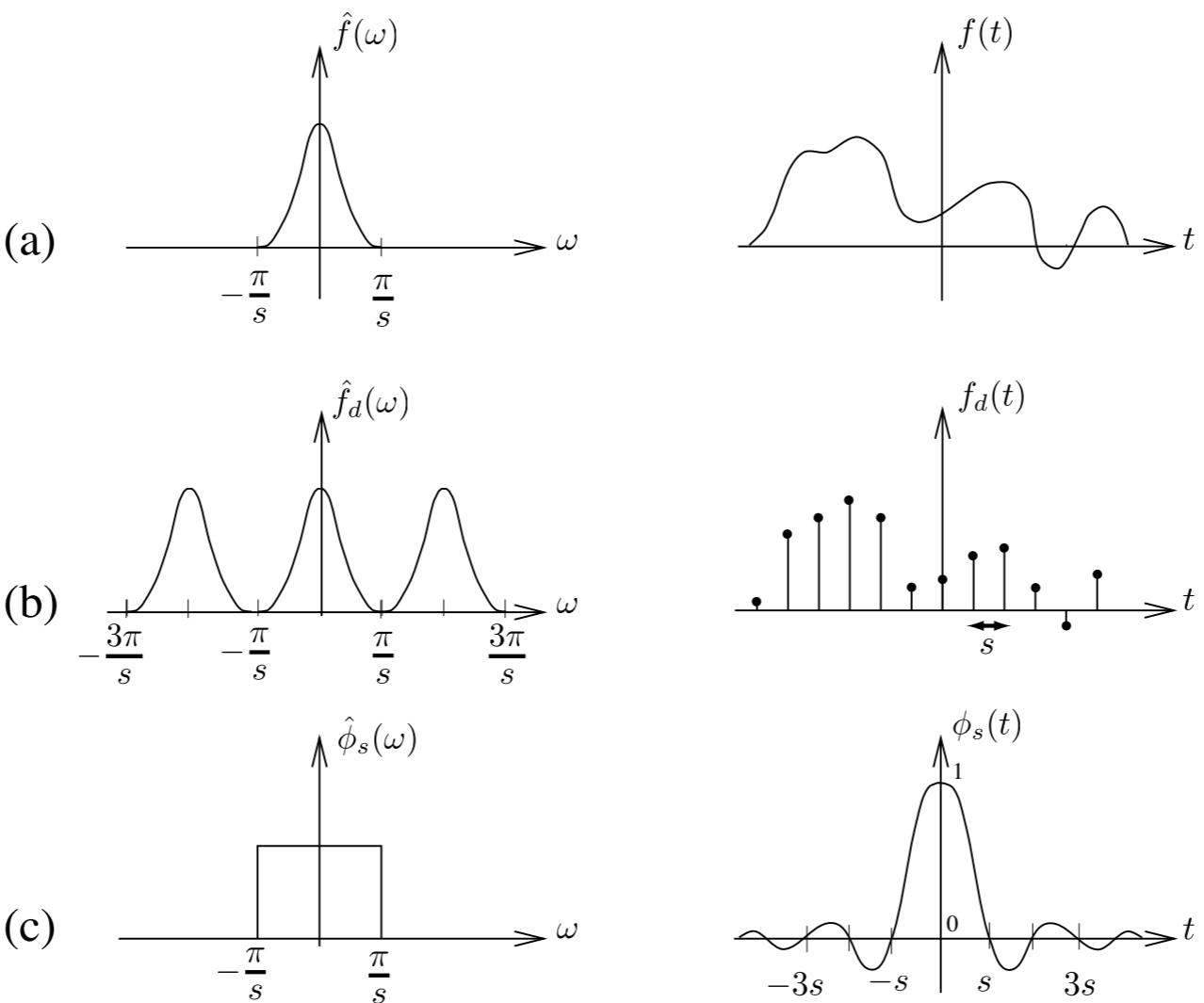
Échantillonnage



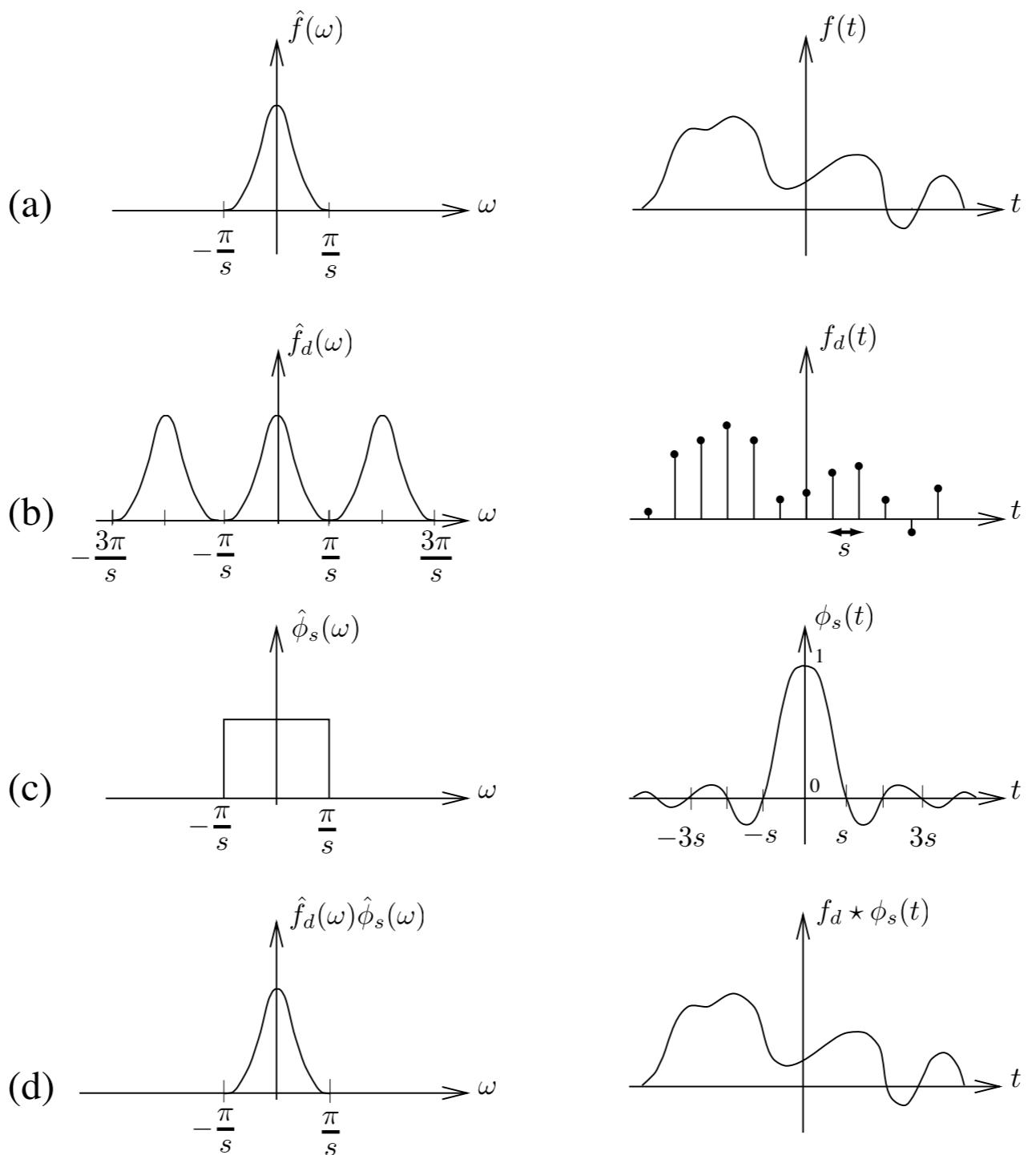
Échantillonnage



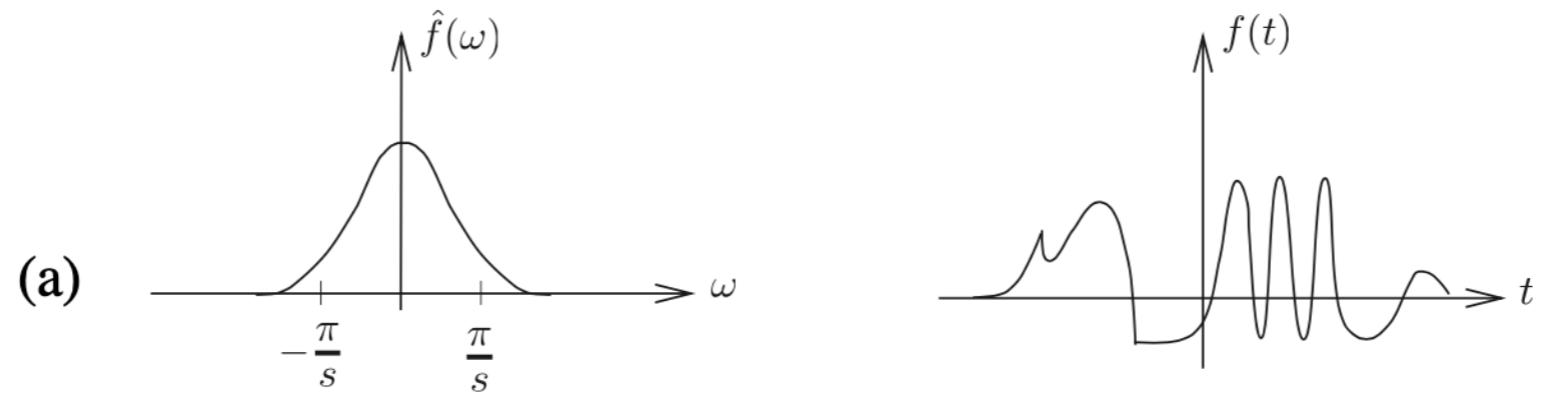
Échantillonnage



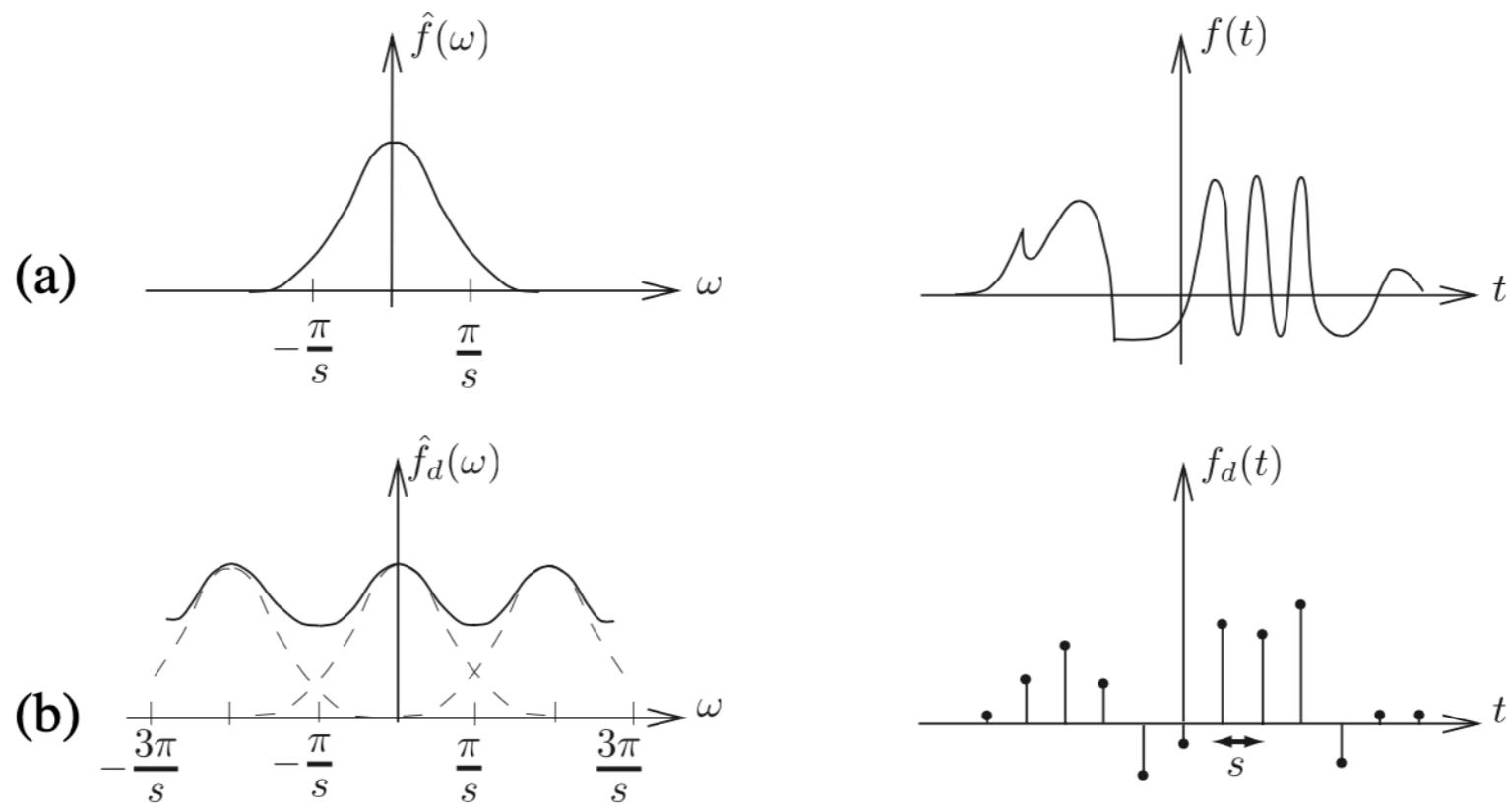
Échantillonnage



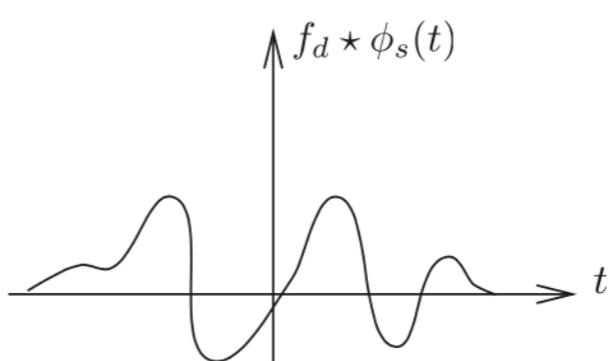
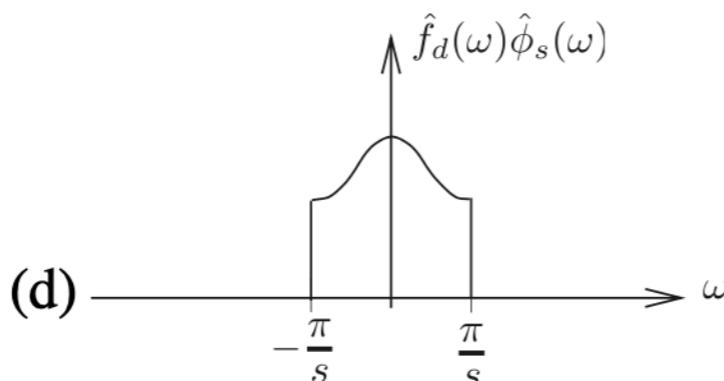
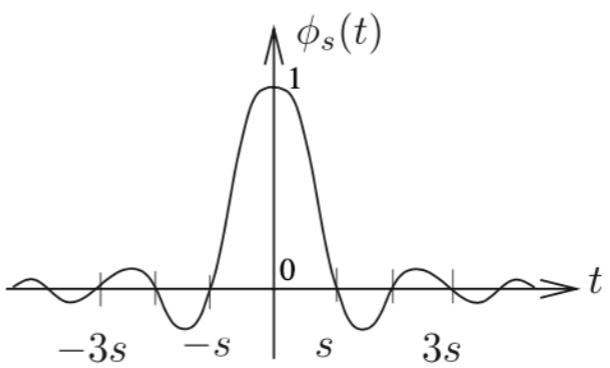
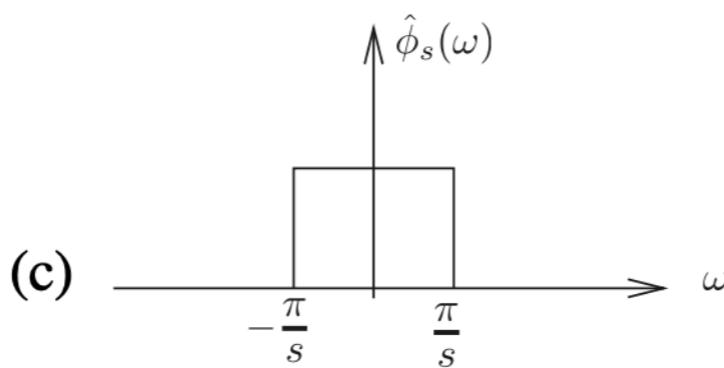
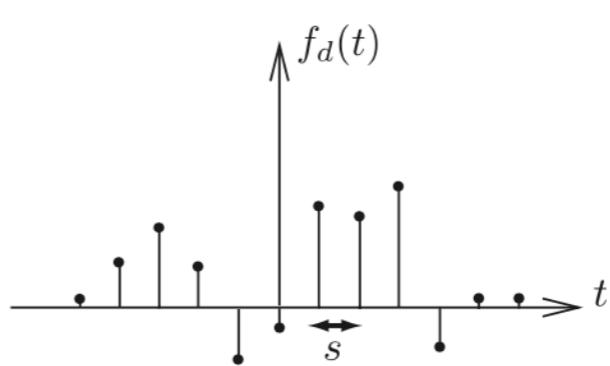
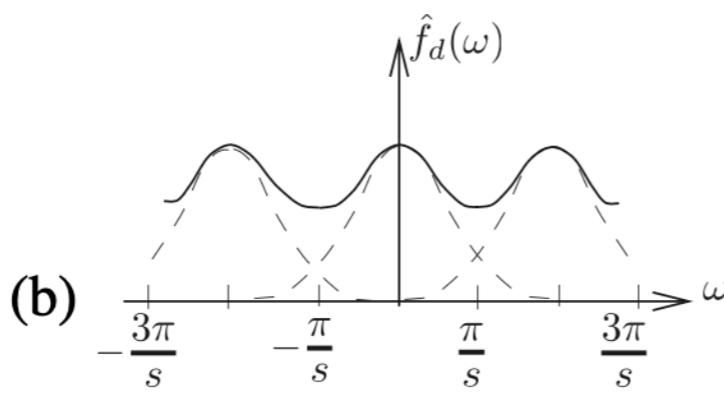
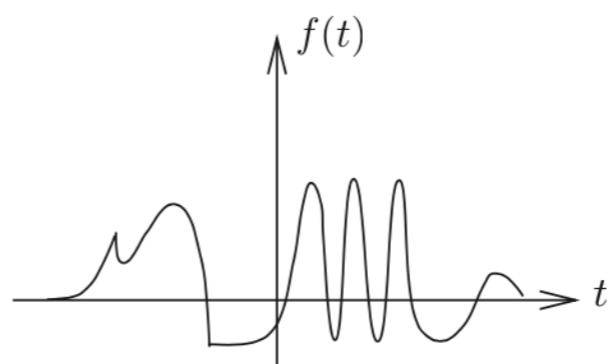
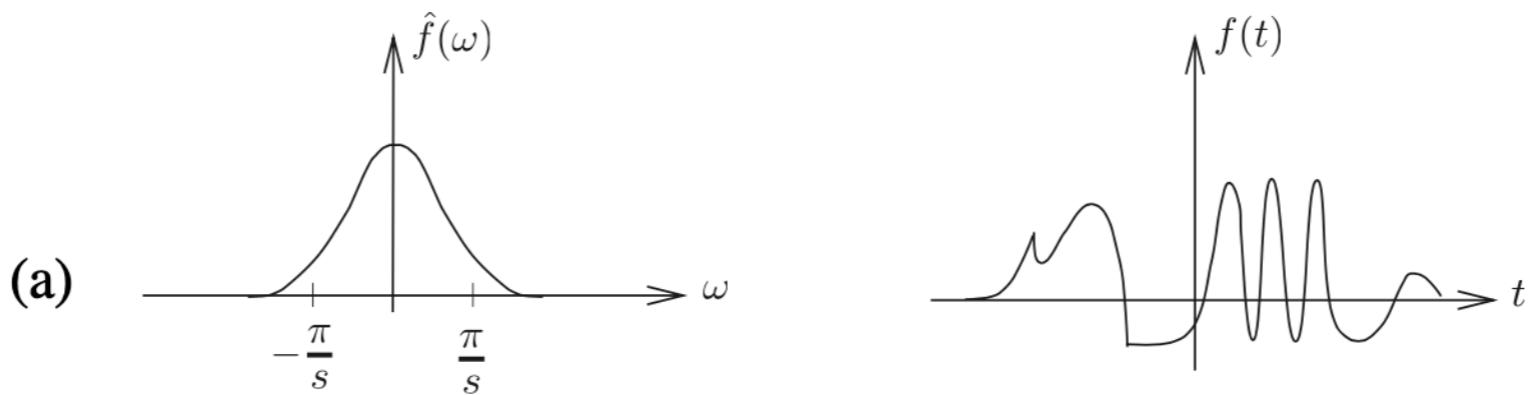
Échantillonnage



Échantillonnage



Échantillonnage

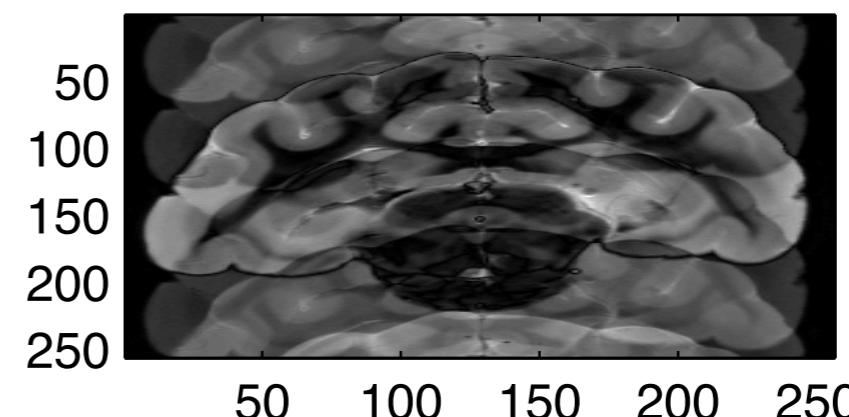
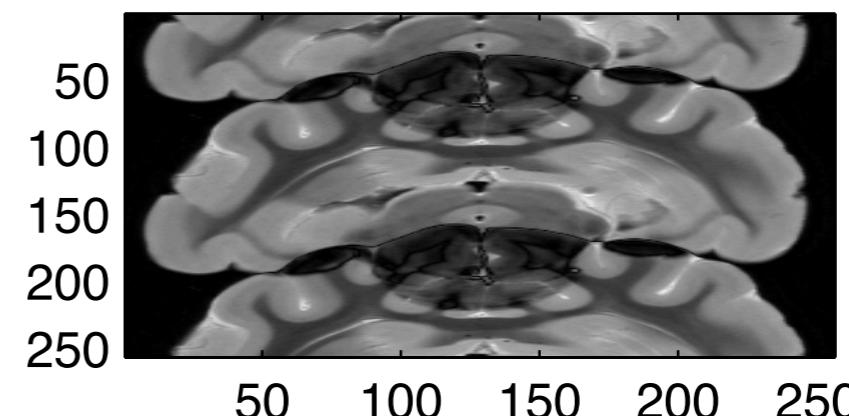
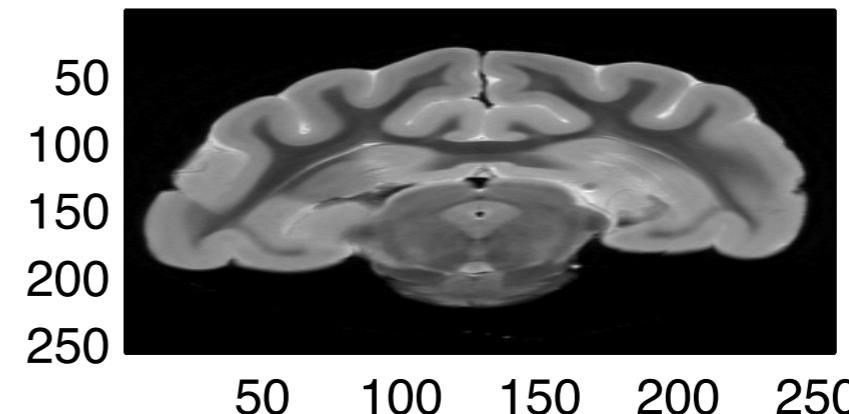
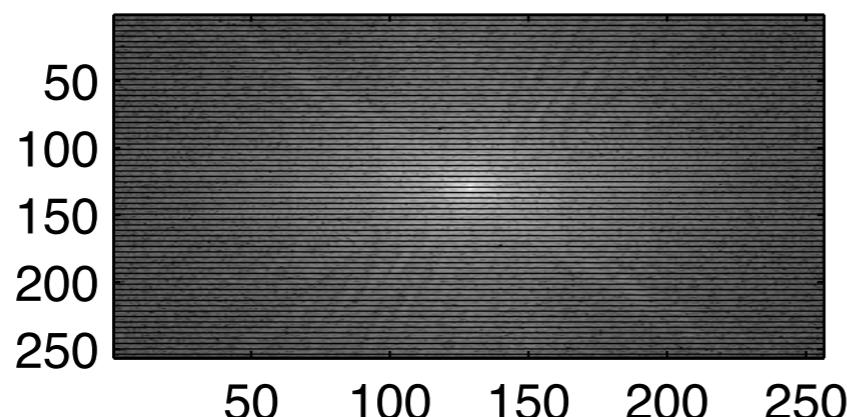
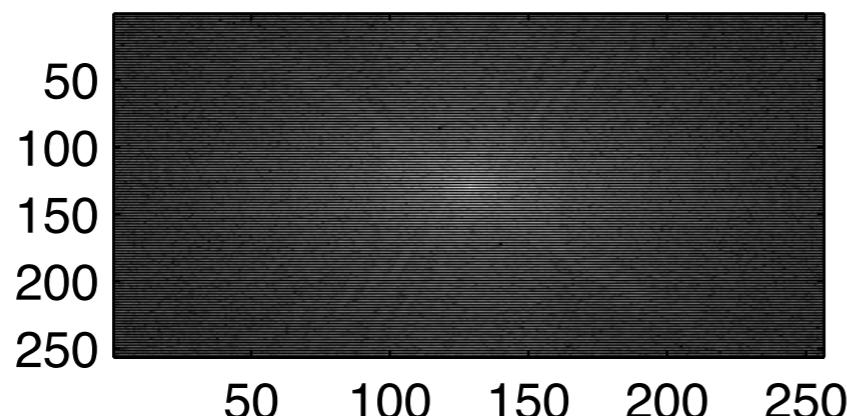
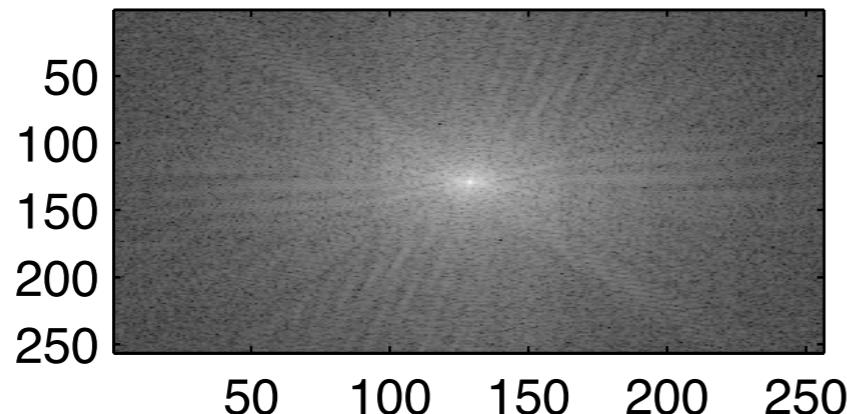


Problème d'échantillonnage

- ⦿ Repliement
- ⦿ Artéfactes de géométriques
- ⦿ Bruit
- ⦿ Interférences

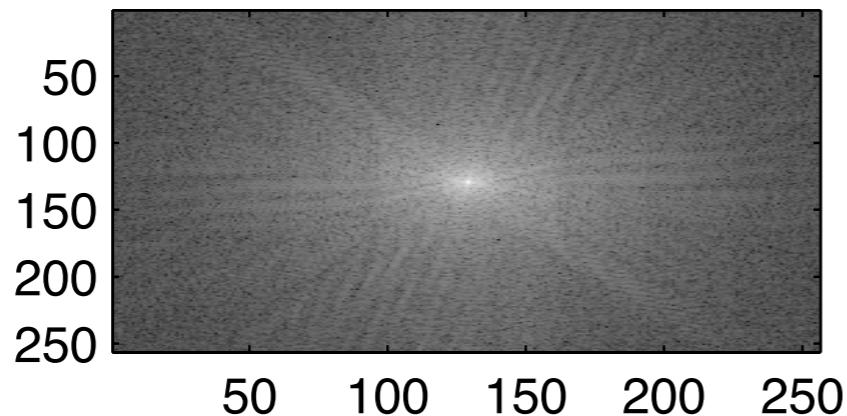
Repliements horizontaux

50%
25%

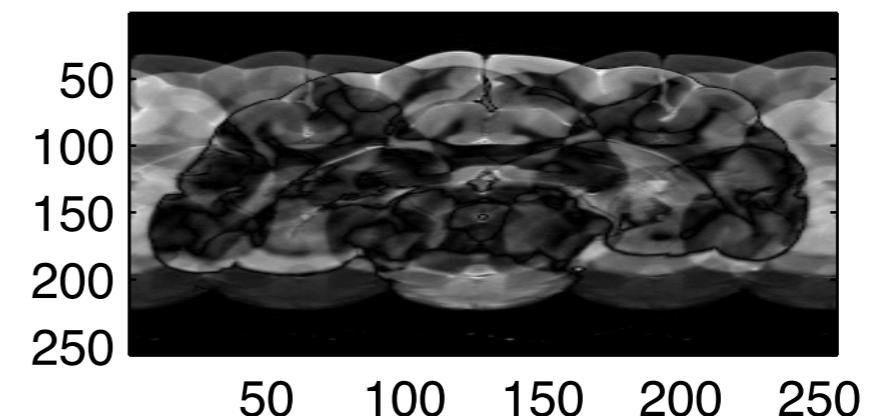
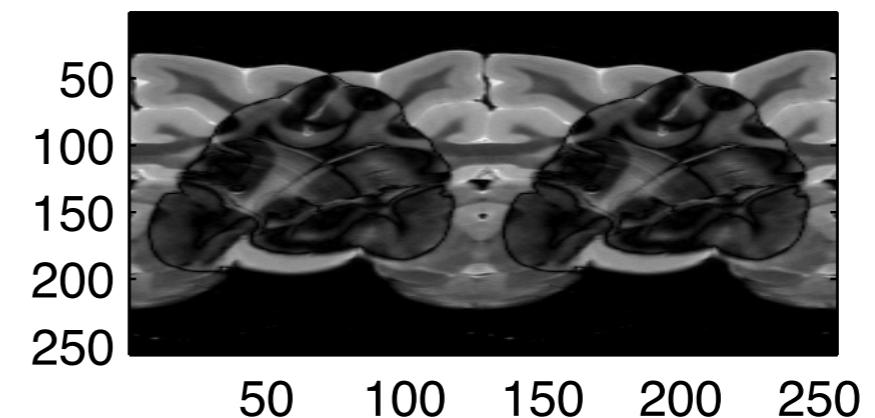
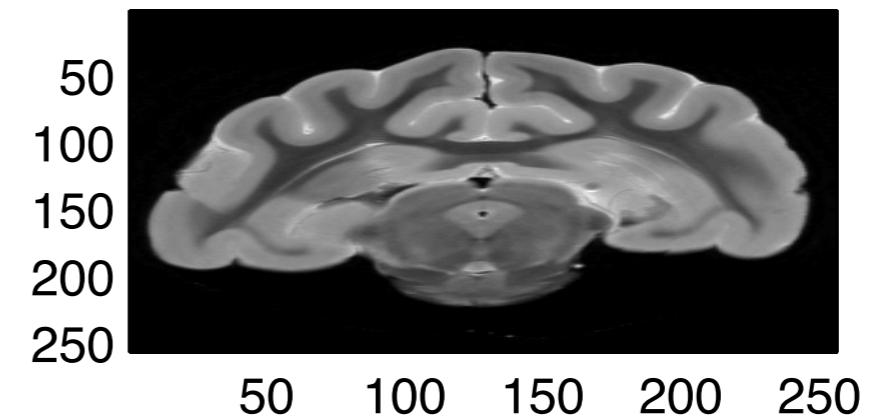
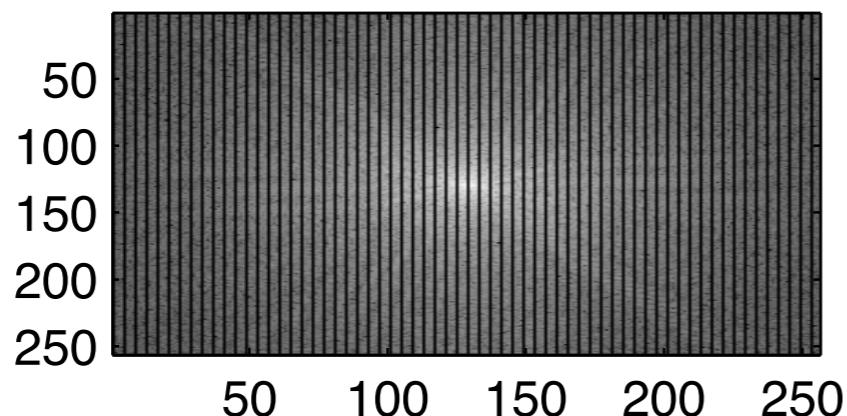


Repliements verticaux

50%



25%



Problème d'échantillonnage

- ⦿ Pourquoi des repliements?
- ⦿ Sauter une rangée ou une colonne sur N équivaut à multiplier par un peigne de Dirac les fréquences (un peigne serré, avec peu d'espace entre les brins)

Problème d'échantillonnage

☞ Pourquoi des repliements?

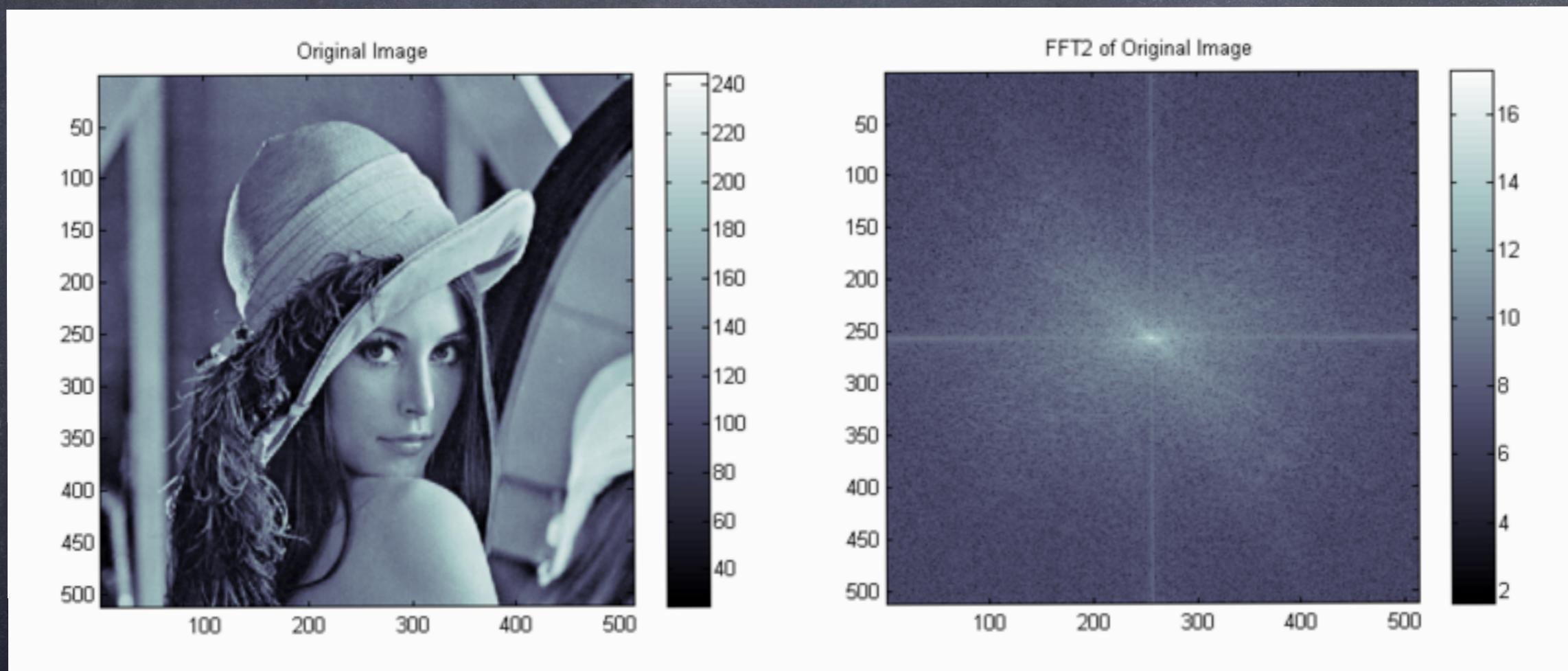
- ☞ Sauter une rangée ou une colonne sur N équivaut à multiplier par un peigne de Dirac les fréquences (un peigne serré, avec peu d'espace entre les brins)
- ☞ C'est donc comme convoluer par un peigne de Dirac dans l'espace très espacé et prendre N copies de l'image originale

Problème d'échantillonnage

☞ Pourquoi des repliements?

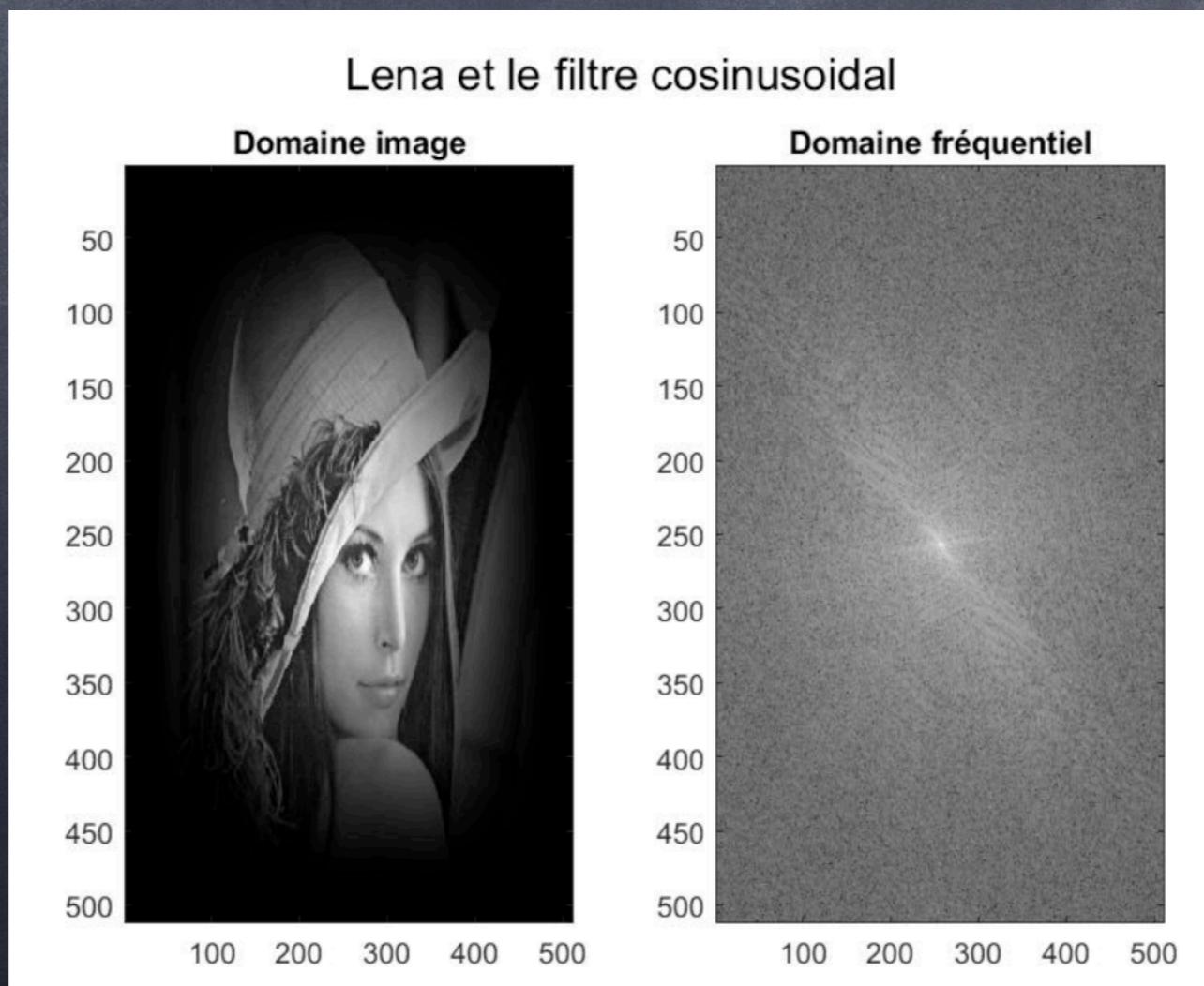
- ☞ Sauter une rangée ou une colonne sur N équivaut à multiplier par un peigne de Dirac les fréquences (un peigne serré, avec peu d'espace entre les brins)
- ☞ C'est donc comme convoluer par un peigne de Dirac dans l'espace très espacé et prendre N copies de l'image originale
- ☞ C'est un peu comme le théorème d'échantillonnage à l'envers

Fourier

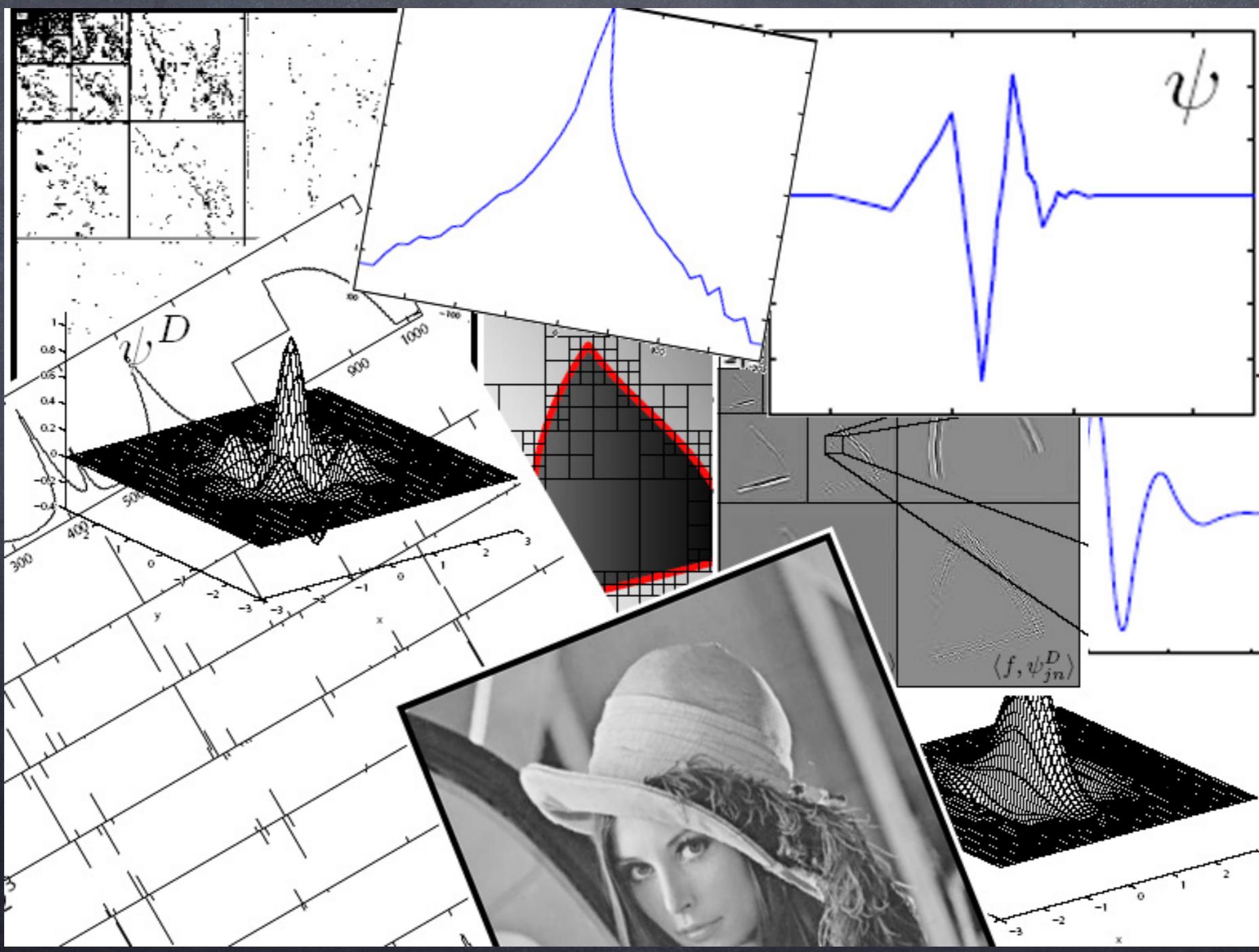


Fourier

- On multipliant par un filtre (e.g. cos) l'image de Lena est atténuée au bords de l'image. Cela enlève les transitions brutes entre l'image et le fond, ce qui demande une infinité de cos et sin pour attraper ces changements
- La fft2 de l'image masquée enlève donc la croix blanche



Au delà de Fourier: Ondelettes



Les transformées

- ⦿ FFT - Fourier
- ⦿ DCT - Cosinus Discrets
- ⦿ DCT locale - Cosinus Discrets Locaux
- ⦿ FWT - Ondelettes
- ⦿ Haar
- ⦿ Daubechies

Rapidité

- ⦿ $O(N \log(N))$
- ⦿ FFT, DCT, DCT locale
- ⦿ $O(N)$
- ⦿ FWT

Mémoire

- ⦿ Toutes $O(N)$ en mémoire
- ⦿ FFT est complexe => $2N$
- ⦿ DCT, DCT locale, FWT => N

Approximations/Compression

- On garde que les m premières fréquences (linéaire) ou les m plus grandes fréquences (non-linéaires)

Original image



Approximations/Compression

- On garde que les m premières fréquences (linéaire) ou les m plus grandes fréquences (non-linéaires)

Original image



$m/n^2 = 0.02$, SNR=19.3dB



2%

Approximations/Compression

- On garde que les m premières fréquences (linéaire) ou les m plus grandes fréquences (non-linéaires)

Original image



$m/n^2=0.02$, SNR=19.3dB



$m/n^2=0.1$, SNR=26.8dB

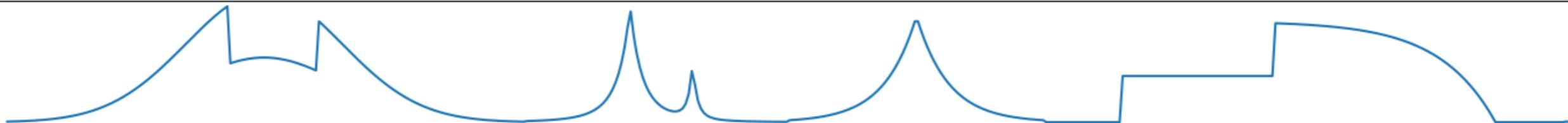


2%

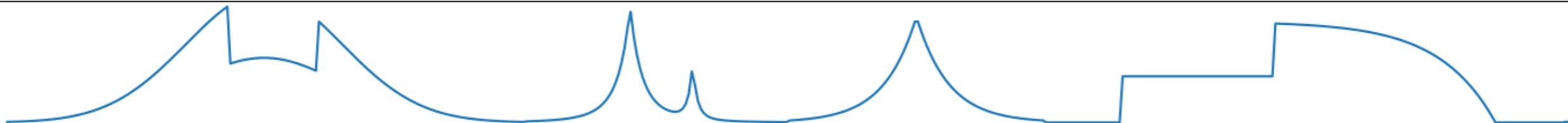
10%

En 1D

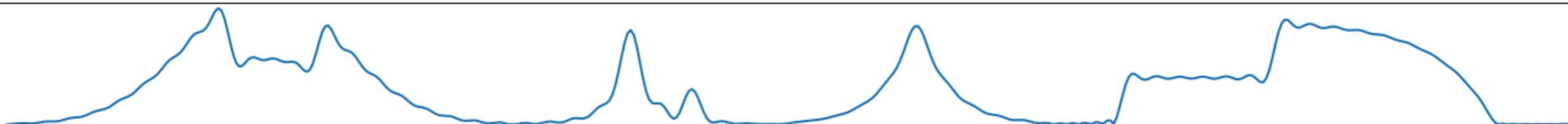
Fourier
Signal Original



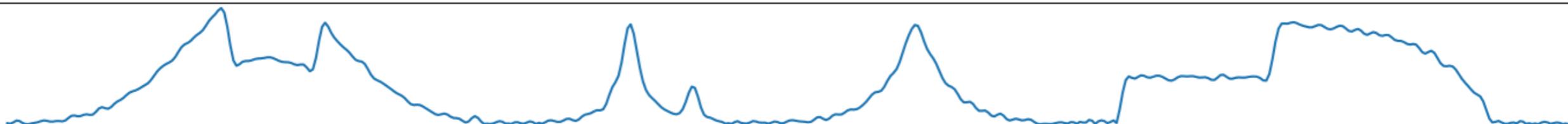
Signal avec Fourier au complet
Erreur relarive = 2.3267965557939184e-16



Approximation linéaire avec 128 coefficients FFT
Erreur relarive = 0.06760493623395143

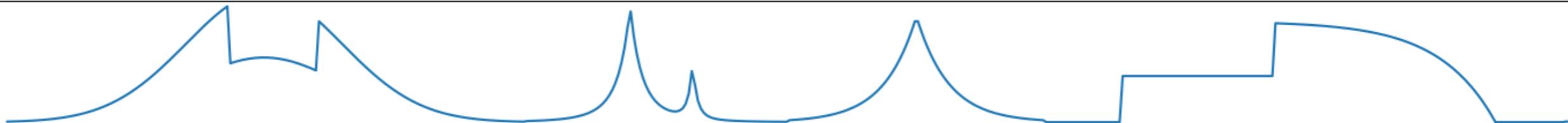


Approximation non-linéaire avec 128 coefficients FFT
Erreur relarive = 0.05856814969370549

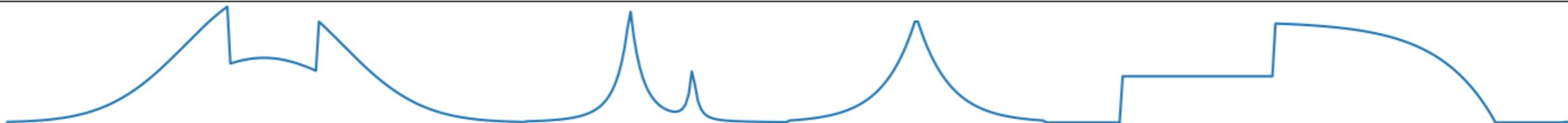


En 1D

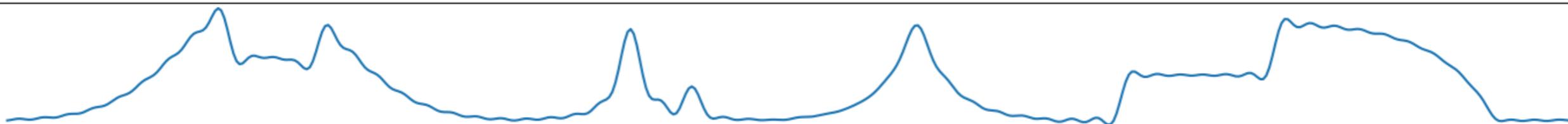
Fourier
Signal Original



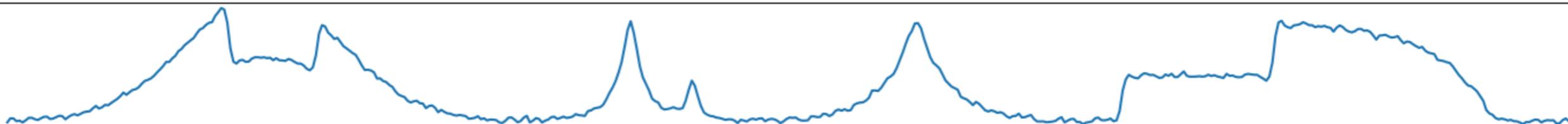
Signal avec DCT au complet
Erreur relative = 2.92041860141912e-16



Approximation linéaire avec 128 coefficients DCT
Erreur relative = 0.06817791491918412

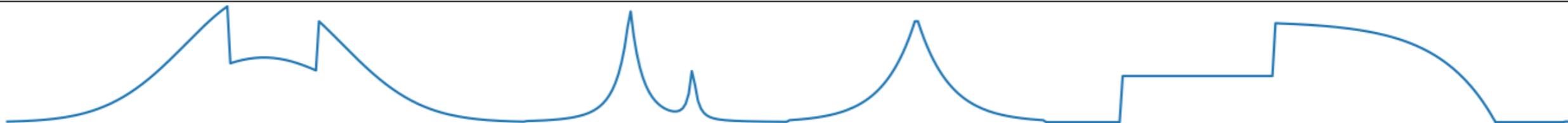


Approximation non-linéaire avec 128 coefficients DCT
Erreur relative = 0.051349817978042306

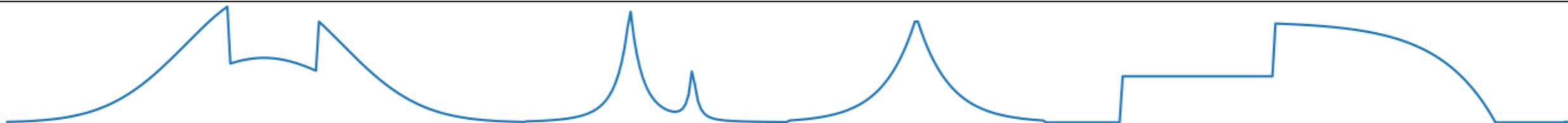


En 1D

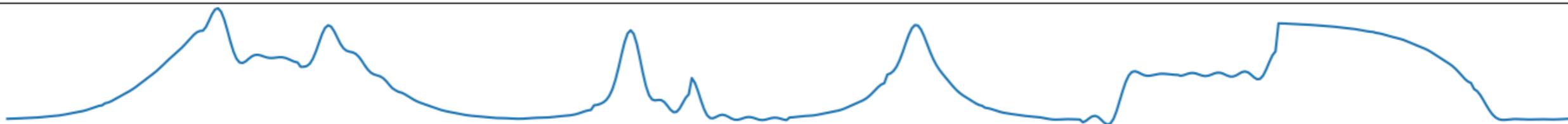
DCT locales
Signal Original



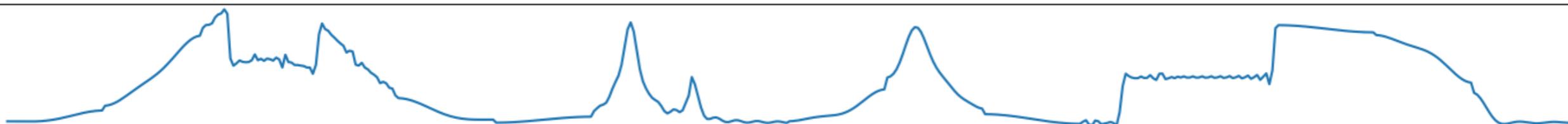
Signal avec la DCT locale au complet
Erreur relarive = 1.4145889249763474e-16



Approximation linéaire avec 128 coefficients DCT locale
Erreur relarive = 0.0724991053564431



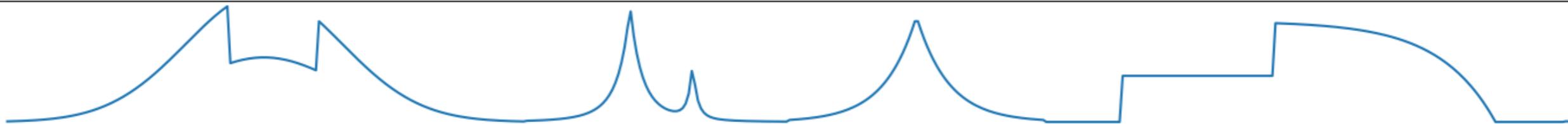
Approximation non-linéaire avec 128 coefficients DCT locale
Erreur relarive = 0.03872875713441316



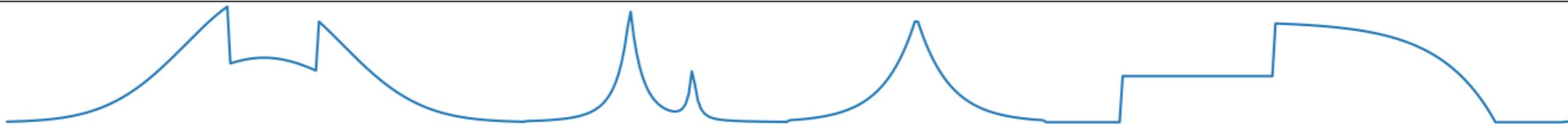
En 1D

Ondelettes de Haar

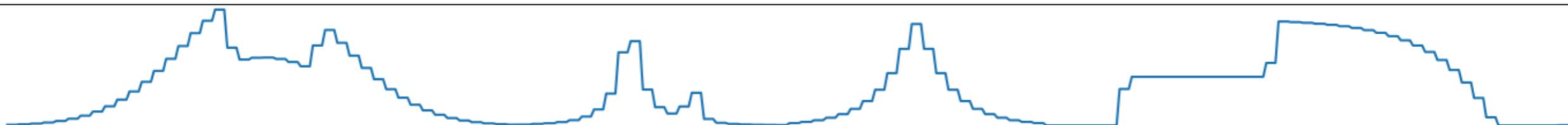
Signal Original



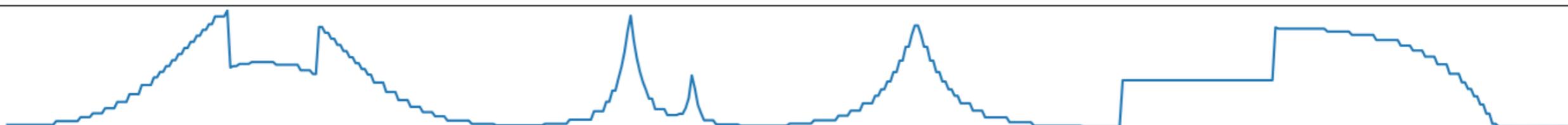
Signal avec Haar au complet
Erreur relarive = 1.4041311182364502e-15



Approximation linéaire avec 128 coefficients de Haar
Erreur relarive = 0.11111481665310408



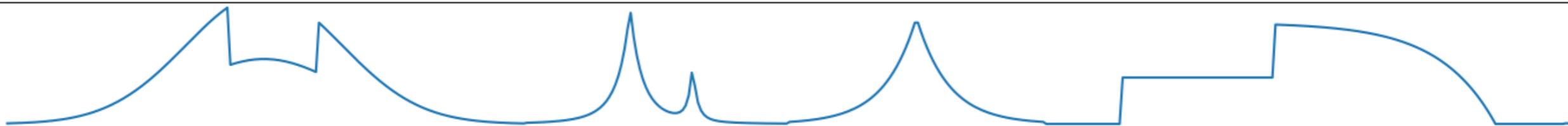
Approximation non-linéaire avec 128 coefficients de Haar
Erreur relarive = 0.029576301205863433



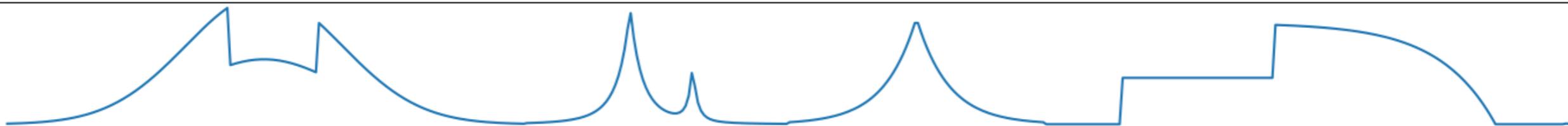
En 1D

Ondelettes D4

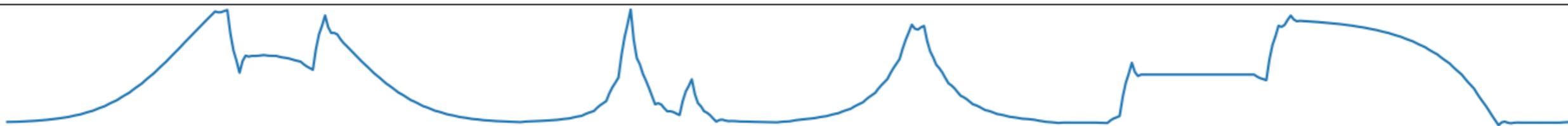
Signal Original



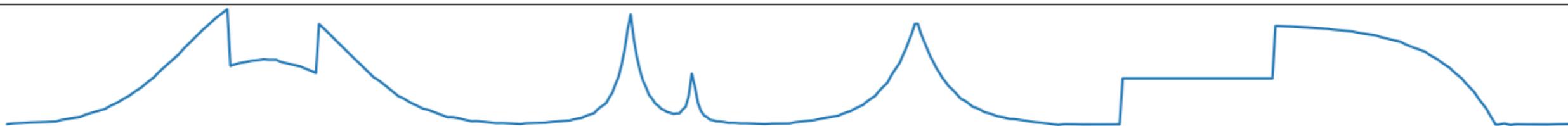
Signal avec D4 au complet
Erreur relarive = 0.00014547174361205677



Approximation linéaire avec 128 coefficients de D4
Erreur relarive = 0.06624713140327726



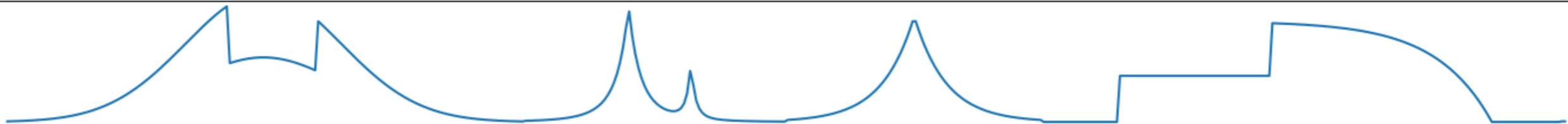
Approximation non-linéaire avec 128 coefficients de D4
Erreur relarive = 0.00652195515483262



En 1D

Ondelettes D4

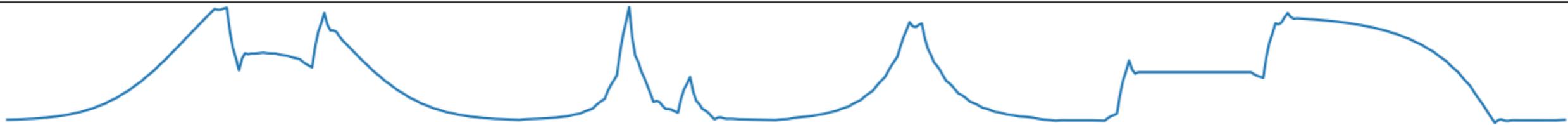
Signal Original



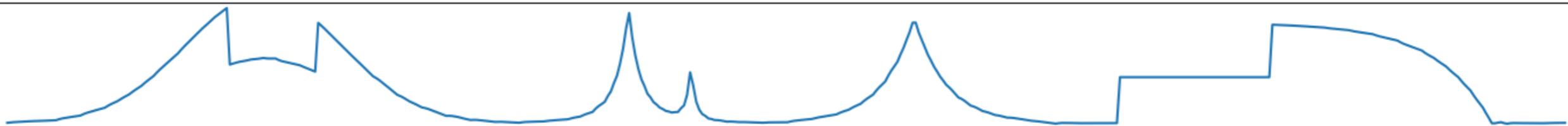
Signal avec D4 au complet
Erreur relarive = 0.00014547174361205677



Approximation linéaire avec 128 coefficients de D4
Erreur relarive = 0.06624713140327726



Approximation non-linéaire avec 128 coefficients de D4
Erreur relarive = 0.00652195515483262

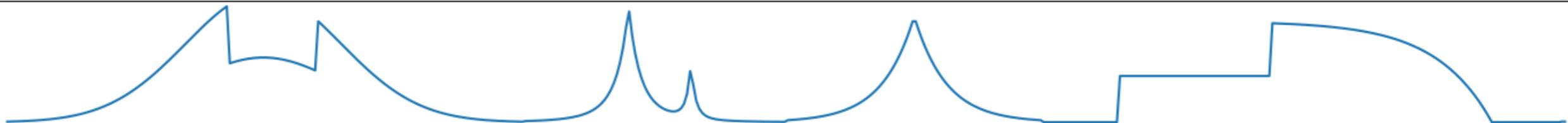


Erreurs de toutes les transformées linéaires ~ 0.07

En 1D

Ondelettes D4

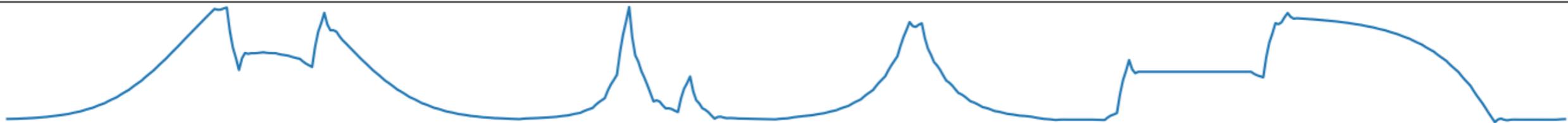
Signal Original



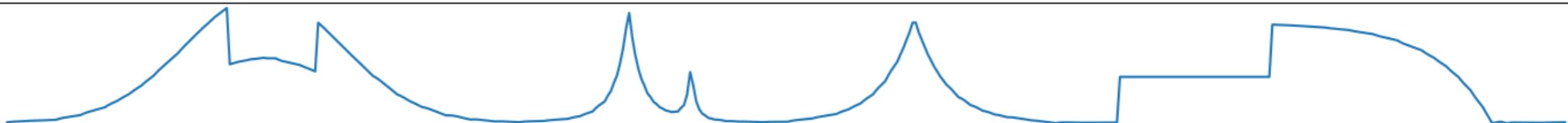
Signal avec D4 au complet
Erreur relarive = 0.00014547174361205677



Approximation linéaire avec 128 coefficients de D4
Erreur relarive = 0.06624713140327726



Approximation non-linéaire avec 128 coefficients de D4
Erreur relarive = 0.00652195515483262



Erreurs de toutes les transformées linéaires ~ 0.07

Erreurs non-linéaires:

FFT $\sim =$ DCT $>$ DCT locale $\sim =$ Haar $>$ D4

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Ça dépend du signal et de l'image!
- ➋ Mais, règle générale ...

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?

Équivalentes

=

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ➋ Fourier vs DCT non-linéaires?

Équivalentes

=

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires? =
- ➋ Fourier vs DCT non-linéaires? =

Équivalentes

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires? =
- ➋ Fourier vs DCT non-linéaires? =
- ➌ Fourier/DCT vs DCT locale non-linéaires?

Équivalentes

=

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ➋ Fourier vs DCT non-linéaires? =
- ➌ Fourier/DCT vs DCT locale non-linéaires?

Équivalentes
=

DCT locale

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ☛ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ☛ Fourier vs DCT non-linéaires? =
- ☛ Fourier/DCT vs DCT locale non-linéaires?
- ☛ Fourier/DCT vs FWT non-linéaires?

Équivalentes
=

DCT locale

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ☛ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ☛ Fourier vs DCT non-linéaires? =
- ☛ Fourier/DCT vs DCT locale non-linéaires?
- ☛ Fourier/DCT vs FWT non-linéaires?

Équivalentes
=

DCT locale

FWT

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ➋ Fourier vs DCT non-linéaires? =
- ➌ Fourier/DCT vs DCT locale non-linéaires?
- ➍ Fourier/DCT vs FWT non-linéaires?
- ➎ DCT locale vs FWT Haar non-linéaires?

Équivalentes
=

DCT locale

FWT

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ➋ Fourier vs DCT non-linéaires? =
- ➌ Fourier/DCT vs DCT locale non-linéaires?
- ➍ Fourier/DCT vs FWT non-linéaires?
- ➎ DCT locale vs FWT Haar non-linéaires?

Équivalentes
=

DCT locale

FWT

DCT locale

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires?
- ➋ Fourier vs DCT non-linéaires? =
- ➌ Fourier/DCT vs DCT locale non-linéaires?
- ➍ Fourier/DCT vs FWT non-linéaires?
- ➎ DCT locale vs FWT Haar non-linéaires?
- ➏ DCT locale vs FWT Daubechies non-linéaires?

Équivalentes
=

DCT locale

FWT

DCT locale

Quelles approximations meilleures?

(demo12 en 1D ou TP4 en 2D)

- ➊ Fourier vs DCT vs
DCT locale vs ondelettes linéaires? Équivalentes
- ➋ Fourier vs DCT non-linéaires? =
- ➌ Fourier/DCT vs DCT locale non-linéaires? DCT locale
- ➍ Fourier/DCT vs FWT non-linéaires? FWT
- ➎ DCT locale vs FWT Haar non-linéaires? DCT locale
- ➏ DCT locale vs FWT Daubechies non-linéaires? FWT

En 2D, exactement comme en 1D?

- ⦿ Non!
- ⦿ Les images ont de la textures, des formes, du contenu plus géométrique qu'en 1D
- ⦿ Ça dépend encore plus de l'image!

Non-linéaire FFT

$m/n^2 = 0.0039$, SNR=16.3dB



$m/n^2 = 0.016$, SNR=19.5dB



$m/n^2 = 0.098$, SNR=25.6dB



Non-linéaire DCT

$m/n^2 = 0.0039$, SNR=17.1dB



$m/n^2 = 0.016$, SNR=20.5dB



$m/n^2 = 0.098$, SNR=27dB



Non-linéaire FWT Haar

$m/n^2 = 0.0039$, SNR=16.4dB



$m/n^2 = 0.016$, SNR=20.3dB

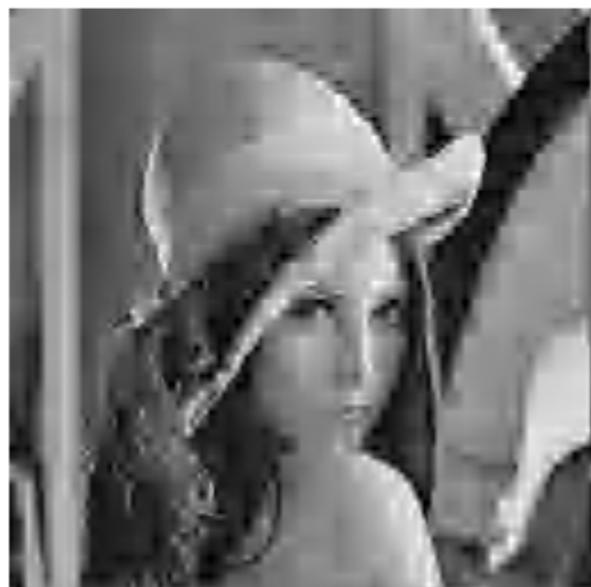


$m/n^2 = 0.098$, SNR=27.9dB



Non-linéaire FWT D4

$m/n^2 = 0.0039$, SNR=17.2dB



$m/n^2 = 0.016$, SNR=21.6dB



$m/n^2 = 0.098$, SNR=29.7dB



Non-linéaire DCT locale

$m/n^2=0.0039$, SNR=14.2dB



$m/n^2=0.016$, SNR=21.9dB



$m/n^2=0.098$, SNR=30.5dB



Non-linéaire FWT Bi-Orthogonal 7-9

$m/n^2 = 0.0039$, SNR=18.2dB



$m/n^2 = 0.016$, SNR=23dB

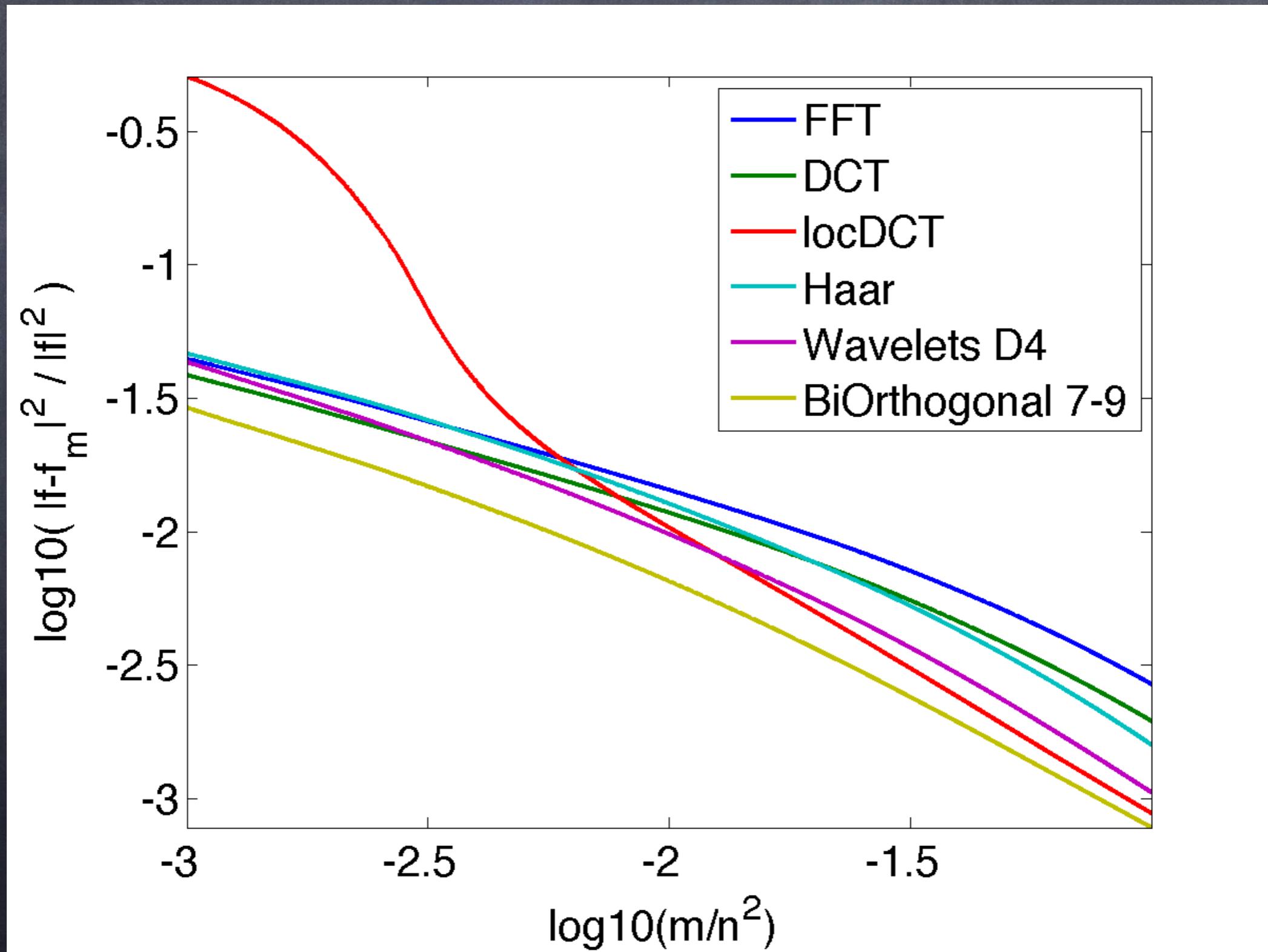


$m/n^2 = 0.098$, SNR=30.9dB



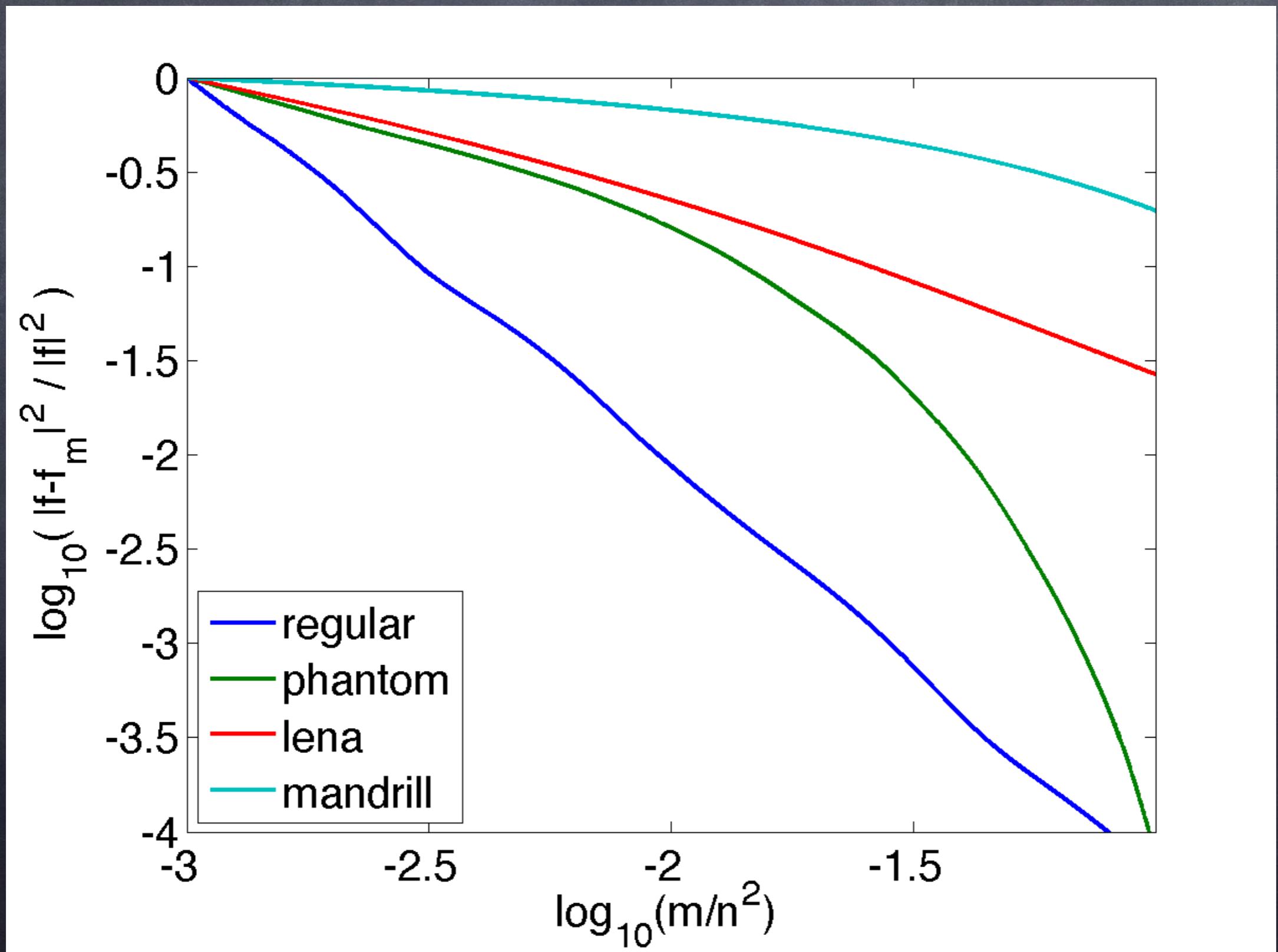
Compression 2D

(plus la courbe est basse, le mieux ça compresse)



Compression 2D

(plus la courbe est basse, le mieux ça compresse)



CORRIGÉ

Fourier 1D et python (Fft, ifft, fftshift)

CORRIGÉ

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

CORRIGÉ

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Convolution

CORRIGÉ

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Convolution

Échantillonnage

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Convolution

Échantillonnage

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Convolution

Multi-résolution

Échantillonnage

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Convolution

Multi-résolution

Échantillonnage

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Ondelettes

Convolution

Multi-résolution

Échantillonnage

Fourier 1D et python (Fft, ifft, fftshift)

Fourier 2D (fft2, ifft2)

Ondelettes

Convolution

Multi-résolution

Échantillonnage

Vrai/Faux

Hi, Dr. Elizabeth?

Yeah, uh... I accidentally took
the Fourier transform of my cat...

