# Filtres gaussiens

## 1. Introduction

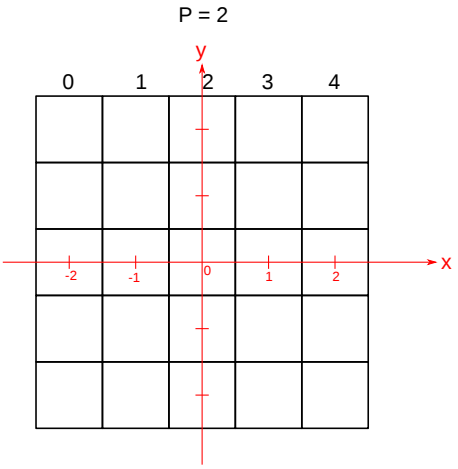
L'objectif de ces TP est de définir et d'utiliser des filtres de convolution *gaussiens* pour le traitement d'image. Ce type de filtre est courrament utilisé dans les logiciels internes des appareils photo numériques, ou dans les logiciels de traitement d'image.

Les applications abordées sont le filtre de lissage pour atténuer le bruit, le filtre dérivateur pour détecter les bords, et la technique d'accentuation de netteté par masque flou.

## 2. Définitions

### 2.a. Masque de convolution

On considère un masque de convolution carré (noté $H$), comportant $2P+1$ lignes et autant de colonnes. Le masque contient la réponse impulsionnelle finie du filtre. Pour calculer les valeurs du masque, on procède à partir d'une fonction $h(x,y)$ qui définit la réponse impulsionnelle infinie du filtre, pour des variables continues $(x,y)$. Le point $(x,y)=(0,0)$ correspond au pixel central du masque, d'indices $(i,j)=(P,P)$. La figure suivante montre le cas $P=2$, qui donne un masque de 5x5 :




### 2.b. Filtre passe-bas

Un filtre passe-bas gaussien est défini à partir de la réponse impulsionnelle suivante :

$$h(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{1}$$

Pour déterminer la valeur de $\sigma$, on choisit une valeur $\epsilon$ petite pour l'élément de $H$ de coordonnées $(x,y)=(P,0)$, situé au centre d'un pixel du bord. La valeur au centre $((x,y)=(0,0))$ est 1. La valeur sur un bord doit être petite par rapport à la valeur au centre. On obtient ainsi :

$$\sigma = \frac{P}{\sqrt{-2\ln(\epsilon)}} \tag{2}$$

La seconde étape est la normalisation de la matrice $H$, qui consiste à diviser tous ses coefficients pour que leur somme soit égale à 1. La normalisation permet de ne pas modifier le niveau de gris moyen de l'image.

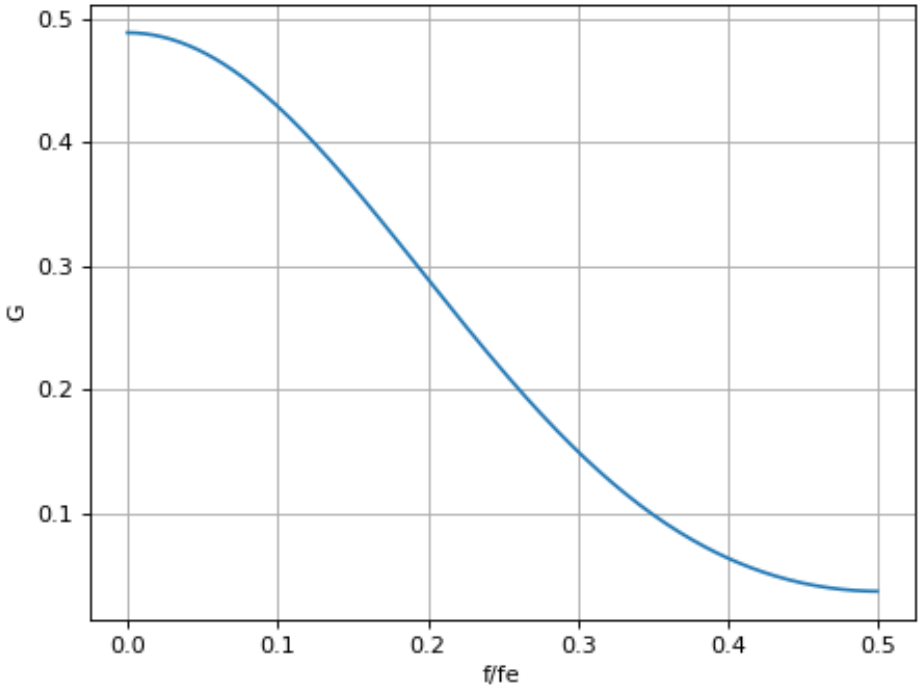
Pour calculer le masque d'un filtre passe-bas, on écrit une fonction `filtrePasseBas(P,epsilon)`, qui renvoie une matrice carrée de taille (2P+1)x(2P+1).

Voici un exemple :

```
P1=2
h1 = filtrePasseBas(P1,0.05)


print(h1)
--> array([[ 0.00059736,  0.00564951,  0.01194726,  0.00564951,  0.00059736],
       [ 0.00564951,  0.05342979,  0.11299024,  0.05342979,  0.00564951],
       [ 0.01194726,  0.11299024,  0.23894527,  0.11299024,  0.01194726],
       [ 0.00564951,  0.05342979,  0.11299024,  0.05342979,  0.00564951],
       [ 0.00059736,  0.00564951,  0.01194726,  0.00564951,  0.00059736]])
```

La réponse fréquentielle sur l'axe X est obtenue comme on le fait pour un signal à une variable :

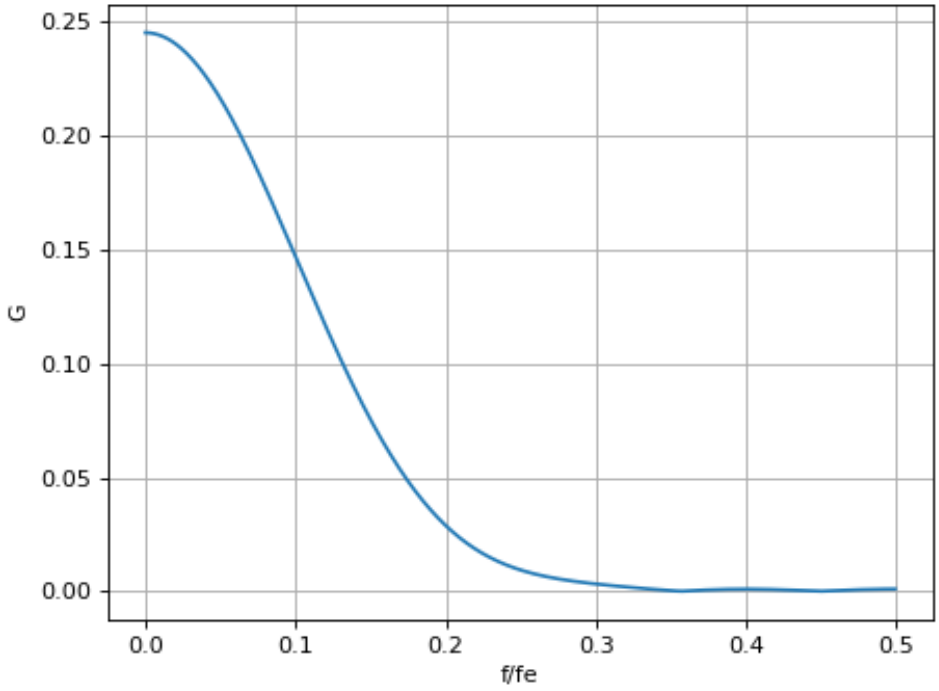
```
import scipy.signal
from matplotlib.pyplot import *

w,g=scipy.signal.freqz(h1[P1])
figure()
plot(w/(2*numpy.pi),numpy.absolute(g))
xlabel("f/fe")
ylabel("G")
grid()
```




La fréquence d'échantillonnage est de 1 par pixel. Un rapport $f/f_e = 0.1$ signifie donc une période de 10 pixels. Pour ce filtre passe-bas, la fréquence de coupure, pour laquelle le gain est la moitié du gain maximal, est environ 0.25, qui correspond à une période de 4 pixels.

Pour abaisser la fréquence de coupure, il faut augmenter la taille du masque :

```
P2=4
h2 = filtrePasseBas(P2,0.05)
w,g=scipy.signal.freqz(h2[P2])
figure()
plot(w/(2*numpy.pi),numpy.absolute(g))
xlabel("f/fe")
ylabel("G")
grid()
```




### 2.c. Filtre dérivateur

Un filtre dérivateur par rapport à $x$ est obtenu en dérivant par rapport à $x$ la fonction gaussienne précédente. À une constante multiplicative près, on obtient :

$$h_x(x, y) = x \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3}$$

En fixant une valeur $\epsilon$ petite pour l'élément $(x, y) = (P, 0)$, on obtient :

$$\sigma = \frac{P}{\sqrt{2(\ln(P) - \ln(\epsilon))}} \tag{4}$$

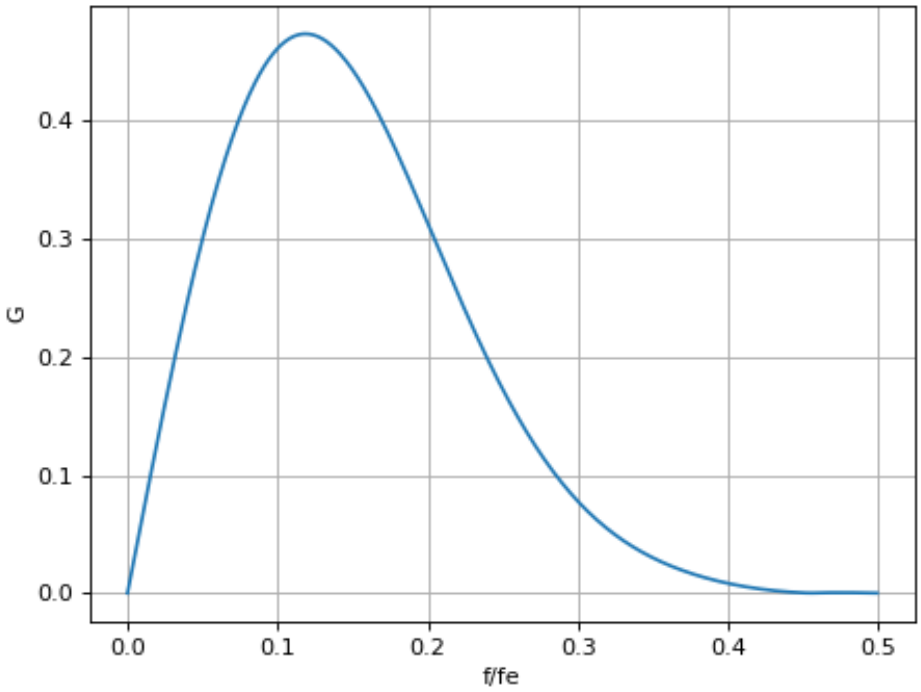
La matrice $H$ comporte une colonne de coefficients nuls ($x = 0$). Les coefficients d'indice $x > 0$ sont positifs ; les coefficients $x < 0$ sont négatifs. La normalisation consiste à diviser les coefficients pour que la somme des coefficients positifs soit égale à 1, et la somme des coefficients négatifs égale à -1.

La fonction `filtreDerivateur(P,epsilon,d)` calcule le masque d'un filtre dérivateur gaussien. La direction de la dérivation est donnée par la liste `d`, qui vaut `[1,0]` ou `[0,1]`.

Voici un exemple, avec sa réponse fréquentielle :

```
P3=4
h3 = filtreDerivateur(P3,0.05,[1,0])
w,g=scipy.signal.freqz(h3[P3])
figure()
plot(w/(2*numpy.pi),numpy.absolute(g))
xlabel("f/fe")
ylabel("G")
```

```
grid()
```

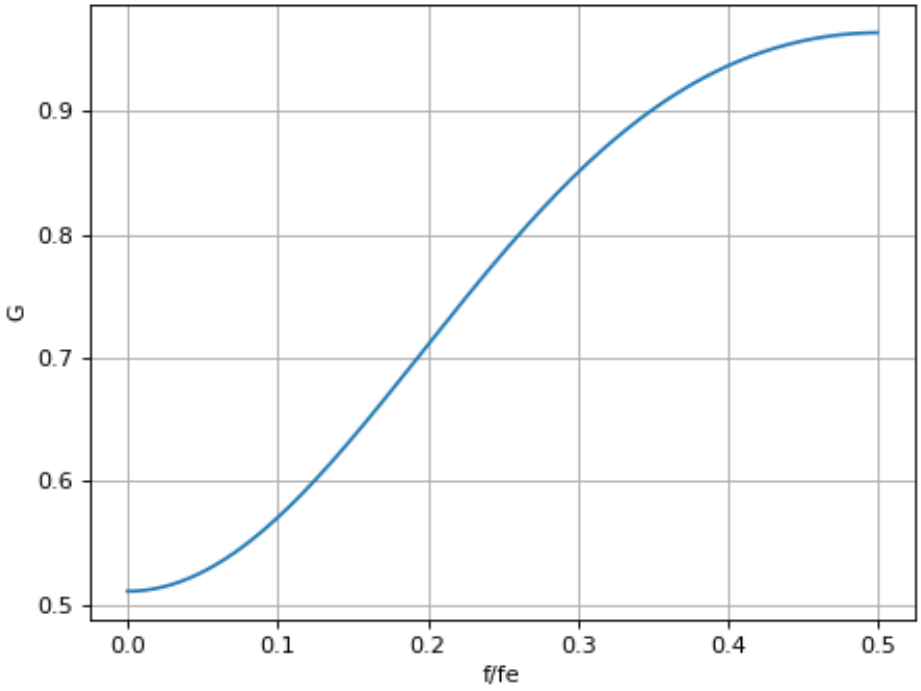



On obtient bien un comportement dérivateur à basse fréquence, associé à un filtrage passe-bas, qui permet de ne pas dériver les variations hautes fréquences, généralement dues au bruit.

### 2.d. Filtre passe-haut et accentuation de netteté

Il est aisé de définir un filtre passe-haut à partir d'un filtre passe-bas, en prenant son complémentaire : on retranche le masque du filtre passe-bas à celui du filtre passe-tout, qui ne change pas la valeur des pixels (matrice H avec un 1 au centre seulement).

```
h4 = -h1
h4[P1,P1] = h4[P1,P1]+1
w,g=scipy.signal.freqz(h4[P1])
figure()
plot(w/(2*numpy.pi),numpy.absolute(g))
xlabel("f/fe")
ylabel("G")
grid()
```




En pratique, il suffit d'appliquer le filtre passe-bas à l'image, et de soustraire l'image obtenue à l'image initiale. C'est la méthode du *masque flou*, utilisée pour augmenter la netteté des images légèrement floues. Le résultat du filtrage passe-haut est ensuite ajouté à l'image initiale, avec un coefficient multiplicateur ajustable, pour augmenter la netteté. Les trois étapes de cette méthode sont donc :

- ▷ Un filtre passe-bas gaussien est appliqué à l'image $A$, pour donner une image floue $B$.
- ▷ L'image floue $B$ est retranchée à l'image $A$ pour donner une image $C$, qui est le résultat d'un filtrage passe-haut sur l'image initiale $A$.
- ▷ L'image initiale est transformée par $D = A + \alpha C$, où $\alpha$ est un coefficient à ajuster en fonction du degré d'accentuation souhaité.

Pour éviter d'accentuer le bruit de l'image, la dernière opération est effectuée sur un pixel seulement si le pixel de l'image $C$ dépasse un seuil. En faisant varier la fréquence de coupure du filtre passe-bas, on peut accentuer l'image à différentes échelles. Les images très légèrement floues nécessitent une accentuation à l'échelle de 1 ou 2 pixels.

## 3. Applications

### 3.a. Outils de traitement des images

Pour lire une image dans un fichier PNG ou JPG on utilise la fonction suivante :

```
A = matplotlib.pyplot.imread("fichier.png")
```

Pour enregistrer une image dans un fichier :

```
matplotlib.pyplot.imsave("fichier.png",A)
```

Dans le cas d'une image en couleur, le tableau A contient trois couches R,G,B. On travaillera sur la couche rouge, que l'on extrait par :

```
rouge = A[:,:,0]
```

Le tableau `rouge` contient des flottants compris entre 0 et 1.

L'application d'un masque de convolution H à une image B se fait de la manière suivante :

```
C = scipy.signal.convolve2d(B,H,mode='same')
```

Le mode `same` permet d'obtenir une matrice C de mêmes dimensions que B. Les pixels situés sur les bords sont traités avec une partie seulement du masque de convolution.

Pour afficher une image en niveau de gris, on utilise la fonction `matplotlib.pyplot.imshow` de la manière suivante :

```
imshow(C,cmap='gray',vmin=0,vmax=1.0)
```

Lorsque les valeurs minimale et maximale ne sont pas précisées, la fonction réalise une transformation affine des valeurs des pixels pour que les niveaux de gris affichés s'étendent du noir jusqu'au blanc.

Pour obtenir la représentation fréquentielle d'une image (transformée de Fourier discrète), on procède de la manière suivante :

```
TFD = numpy.fft.fft2(C)
imshow(numpy.absolute(numpy.fft.fftshift(TFD)),cmap='gray',extent=[-0.5,0.5,-0.5,0.5]
xlabel("fx/fe")
ylabel("fy/fe")
```

### 3.b. Lissage d'une image

Le lissage d'une image consiste à appliquer un filtrage passe-bas de manière à atténuer le contraste des détails les plus fins (à l'échelle de quelques pixels). Différentes raisons peuvent motiver ce type de traitement :

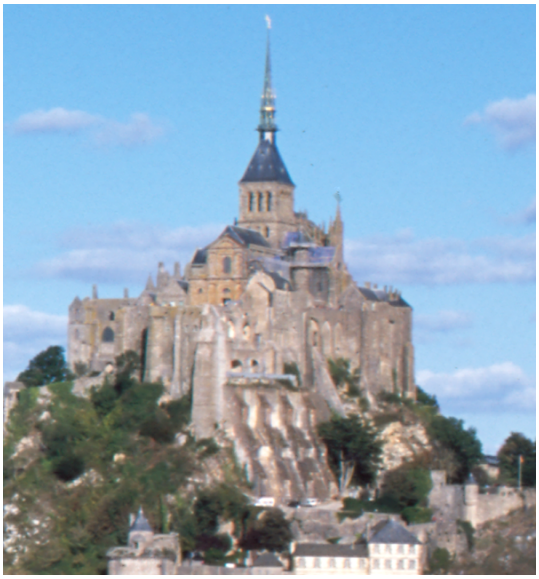
- ▷ Réduction du bruit.
- ▷ Réduction des hautes fréquences avant d'appliquer un traitement (comme la dérivation) qui au contraire accentue les hautes fréquences.
- ▷ Réduction de la netteté d'une photographie, pour des raisons esthétiques.
- ▷ Réduction de la netteté d'une image de synthèse, pour simuler un flou de mise au point.

Ci-dessous l'image à traiter :

Implémenter la fonction `filtrePasseBas(P,epsilon)` qui fournit le masque d'un filtre passe-bas gaussien.
Appliquer le filtrage à la couche rouge de l'image et afficher le résultat. Essayer plusieurs tailles de masque (P=2,3,etc.) et interpréter les résultats à l'aide de la réponse fréquentielle du filtre.

### 3.c. Accentuation de netteté

Voici l'image à traiter :



Il s'agit d'un détail d'une image obtenue avec un scanner à partir d'une photographie sur film. L'image souffre d'un défaut de netteté à cause du manque de résolution de la partie optique du scanner. Il s'agit d'appliquer la technique du masque flou pour accentuer la netteté de l'image. On travaillera dans un premier temps sur sa couche rouge, avant de traiter les trois couches et d'enregistrer le résultat dans un fichier.

Appliquer un filtre passe-bas gaussien pour obtenir à partir de l'image A une image B floue.
Obtenir le résultat du filtrage passe-haut par la différence $C = A - B$. Afficher cette image.
Appliquer la transformation $D = A + \alpha C$ et observer le résultat.
Fixer $\alpha = 1$ et faire varier la fréquence de coupure du filtre passe-bas (en faisant varier la taille du masque) jusqu'à obtenir un résultat visuellement satisfaisant.

Écrire une fonction qui calcule la somme précédente pixel par pixel, en sommant seulement si la valeur absolue du pixel de C dépasse un seuil. La fonction devra aussi procéder à un écrêtage en cas de valeur finale inférieure à 0 ou supérieure à 1.
Écrire une fonction qui effectue l'accentuation de netteté d'une image en couleur et enregistre le résultat dans un fichier. Les paramètres sont la taille P du masque de filtrage, le coefficient $\alpha$, et le seuil.

### 3.d. Filtre dérivateur

Implémenter la fonction `filtreDerivateur(P,epsilon,d)` qui calcule le masque d'un filtre dérivateur gaussien.
Appliquer ce filtre à la couche rouge de l'image des objets, avec plusieurs tailles de masque.
Comparer au filtre dérivateur de Sobel défini dans Détection des bords.
Calculer et afficher la norme du gradient de la couche rouge.

## 4. Solution

solution-FiltresGaussiens.py