# Conception d'un filtre par placement des zéros et des pôles

## 1. Introduction

Ce document montre comment un filtre numérique récursif peut être conçu en plaçant ses zéros et ses pôles. Cette méthode est très efficace pour concevoir des filtres dont la fréquence de coupure est très basse (une fraction infime de la fréquence d'échantillonnage). On verra en particulier comment obtenir un filtre passe-bas ou passe-bande avec un domaine intégrateur très large, s'étendant d'une fréquence très faible jusqu'à la fréquence de Nyquist.

## 2. Principe

On considère un filtre numérique linéaire défini par la relation de récurrence suivante :

$$y_n = \sum_{k=0}^{N} b_k x_{n-k} - \sum_{k=1}^{M} a_k y_{n-k} \tag{1}$$

Les $M+1$ coefficients $a_n$ et les $N+1$ $b_n$ sont réels.

Pour étudier la réponse fréquentielle du filtre, on introduit la variable complexe Z définie par :

$$z = \exp(i2\pi f T_e) = \exp(i\Omega) \tag{2}$$

où $f$ est la fréquence, $T_e$ la période d'échantillonnage. Pour simplifier les notations, on utilise aussi la pulsation réduite $\Omega$. Lorsque la fréquence varie entre 0 et la fréquence de Nyquist (moitié de la fréquence d'échantillonnage), la pulsation $\Omega$ varie entre 0 et $\pi$. Le nombre $Z$ se déplace donc sur le demi-cercle unité.

La réponse fréquentielle du filtre est obtenue avec la fonction de transfert en Z :

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}} \tag{3}$$

Pour une démonstration de ce résultat, voir Conception et mise en œuvre des filtres numériques.

On peut écrire la fonction de transfert comme un rapport de deux polynômes :

$$H(z) = \frac{b_0 z^N + b_1 z^{N-1} + b_2 z^{N-1} + \cdots b_N}{z^M + a_1 z^{M-1} + a_2 z^{M-2} + \cdots + a_M} \tag{4}$$

Les zéros sont les $N$ racines du numérateur ; on les notera $q_i$. Les pôles sont les racines du dénominateur ; on les notera $p_i$. Pour que le filtre soit stable, il faut et il suffit que tous les pôles soient à l'intérieur (strictement) du cercle unité. Les coefficients du filtre étant réels, chaque pôle (ou zéro) non réel est associé à son complexe conjugué.

Voici par exemple une fonction de transfert bi-quadratique ($M = N = 2$) écrite avec ses pôles et zéros :

$$H(z) = b_0 \frac{(z - q_1)(z - q_2)}{(z - p_1)(z - p_2)} \tag{5}$$

La méthode de placement des pôles ([1],[2]) consiste à définir le filtre en plaçant directement les pôles et les zéros.

Supposons que l'on souhaite annuler $H$ pour une certaine pulsation $\Omega_a$ non nulle. Il faut définir un zéro de module 1 et d'argument $\Omega_a$, et son conjugué :

$$q_1 = \exp(i\Omega_a) \tag{6}$$
$$q_2 = \exp(-i\Omega_a) \tag{7}$$

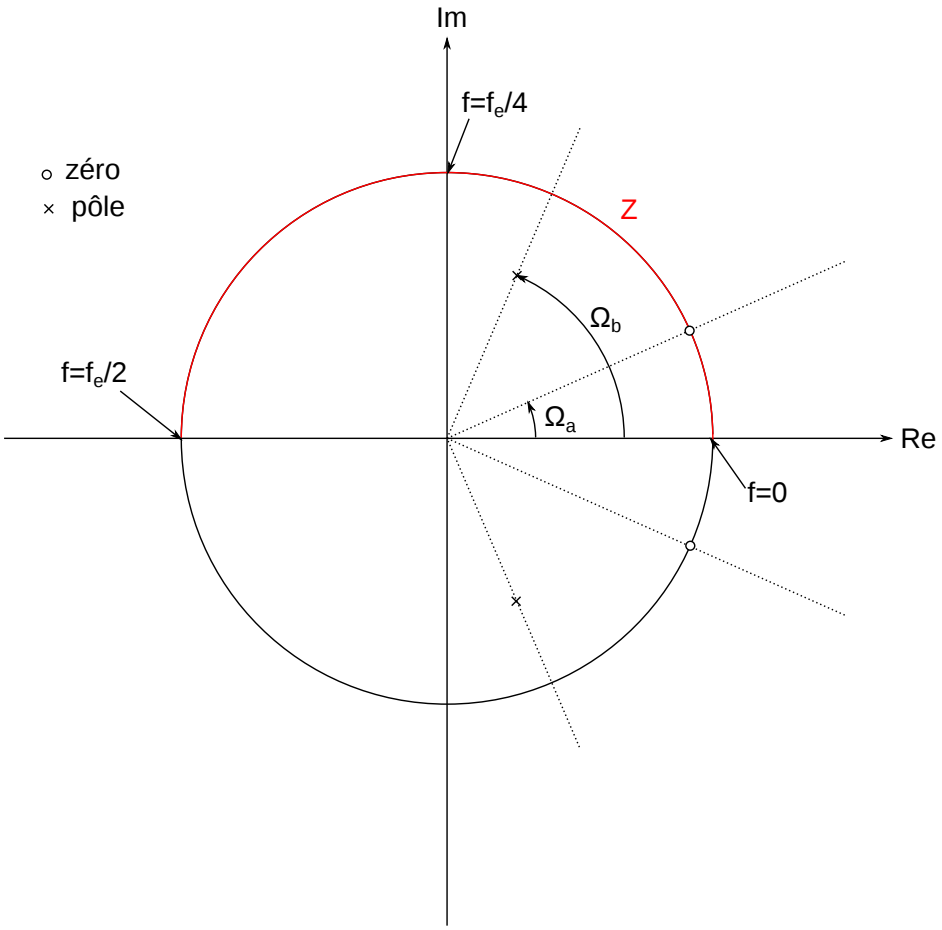
Si la pulsation est nulle, un seul zéro réel suffit. Si l'on veut obtenir pour cette pulsation un gain faible mais non nul, il suffit de donner à ce zéro un module différent de 1 (inférieur ou supérieur à 1).

Pour définir une résonance, c'est-à-dire un maximum du gain, il faut agir sur les pôles. Supposons que l'on souhaite obtenir une résonance pour une pulsation $\Omega_b$. Si cette pulsation est non nulle, il faut définir deux pôles conjugués :

$$p_1 = r_1 \exp(i\Omega_b) \tag{8}$$
$$p_2 = r_1 \exp(-i\Omega_b) \tag{9}$$

Si $\Omega_b = 0$, un seul pôle réel suffit. Le module $r_1$ doit être strictement inférieur à 1 pour que le filtre soit stable. Plus ce module est proche de 1, plus la résonance est forte (plus le maximum est haut). On peut dans certains cas choisir un module égal à 1, ce qui donne une instabilité pour la pulsation $\Omega_b$ (le gain tend vers l'infini pour cette pulsation).

Les pôles et les zéros sont représentés graphiquement dans le plan complexe. Les pôles sont réprésentés par des croix, les zéros par des cercles.



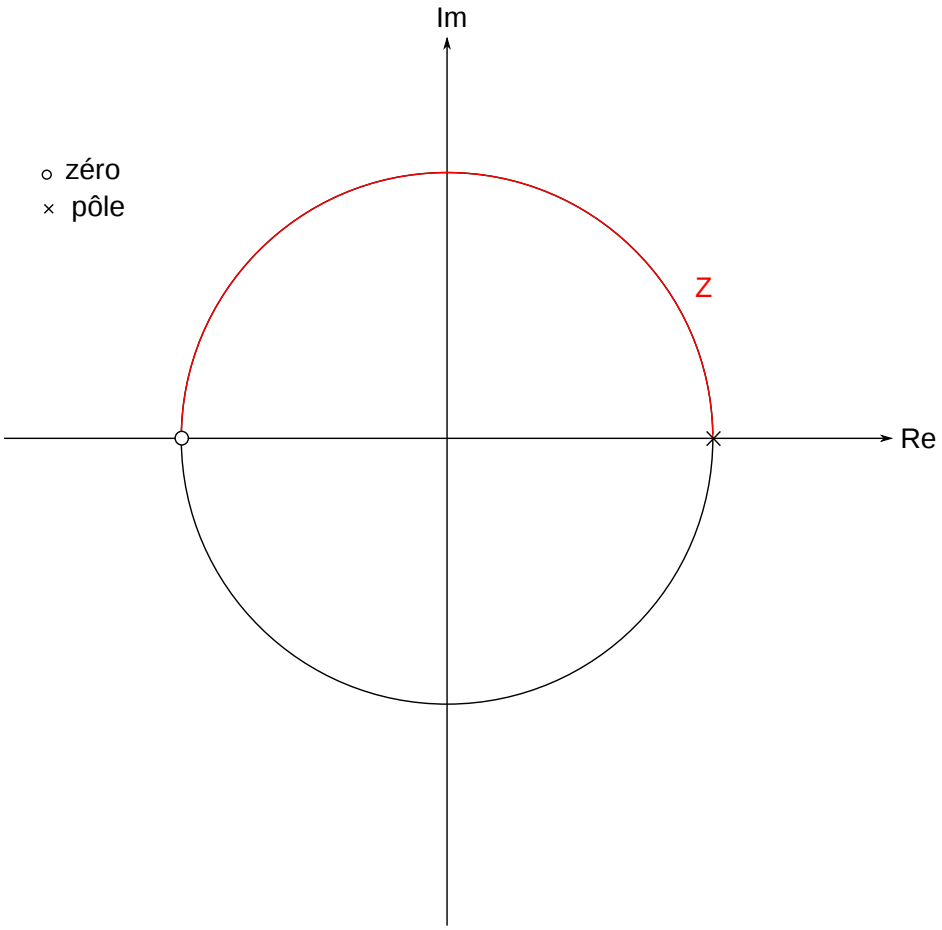

Lorsque les pôles et les zéros sont définis, il faut déterminer la constante $b_0$ en fonction du gain souhaité, par exemple le gain maximal pour un filtre passe-bas.

La méthode de placement des pôles et zéros est une méthode qualitative. Il faut tracer la réponse fréquentielle pour obtenir le comportement quantitatif du filtre.

## 3. Filtres du premier ordre

### 3.a. Intégrateur

Un intégrateur parfait est défini avec un pôle à fréquence nulle $p_1 = 1$ et un zéro à la fréquence de Nyquist $q_1 = -1$ :




$$H(z) = \frac{z+1}{z-1} = \frac{1+z^{-1}}{1-z^{-1}} \tag{10}$$

La seconde écriture permet d'obtenir la relation de récurrence, sachant qu'au numérateur $z^{-1}$ correspond à un retard d'une unité pour l'entrée, au dénominateur un retard d'une unité pour la sortie :

$$y_n = y_{n-1} + x_n + x_{n-1} \tag{11}$$

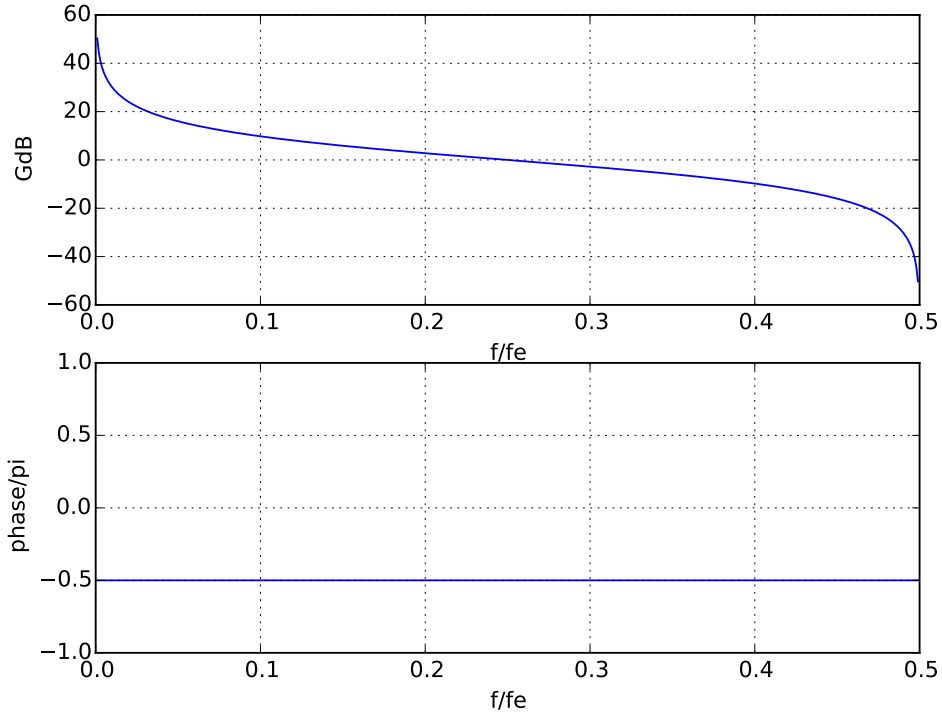
La réponse fréquentielle est :

$$H(\Omega) = \frac{e^{i\Omega}+1}{e^{i\Omega}-1} \tag{12}$$

Le gain est infini à pulsation nulle, ce qui signifie que le filtre est instable si le signal d'entrée possède une fréquence nulle dans son spectre. L'instabilité vient de la présence d'un pôle sur le cercle unité. Voici le tracé du gain et du déphasage :

```
import numpy
import scipy.signal
from matplotlib.pyplot import *
b=[1,1]
```

```
a=[1,-1]
w,h=scipy.signal.freqz(b,a)

figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```




Pour ce filtre, il n'y a pas de gain maximal pour fixer la constante $b_0$. On fixera cette constante en fonction de l'objectif.

Supposons que l'on cherche à réaliser une intégration vraie, qui correspond à la relation en temps continu suivante :

$$y(t) = \int_0^t x(t)\,dt \tag{13}$$

Dans ce cas, la relation de récurrence est

$$y_n = y_{n-1} + \frac{T_e}{2}(x_n + x_{n-1}) \tag{14}$$

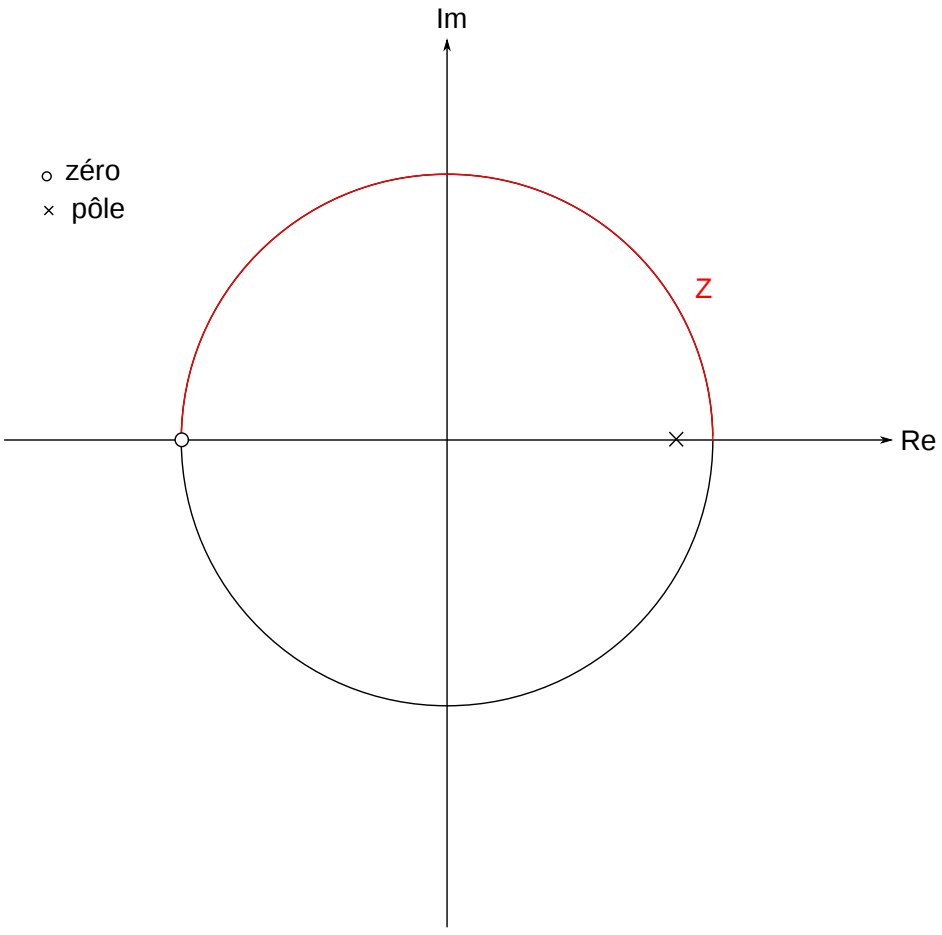
qui correspond à la méthode des trapèzes pour le calcul numérique d'une intégrale. Sa fonction de transfert est :

$$H(z) = \frac{T_e}{2}\frac{z+1}{z-1} = \frac{T_e}{2}\frac{1+z^{-1}}{1-z^{-1}} \tag{15}$$

Le filtre intégrateur joue un rôle important dans la conception des filtres, car il sert de référence dans la méthode de transformation bilinéaire, expliquée dans Filtres à réponse impulsionnelle infinie.

### 3.b. Filtre passe-bas

En partant du filtre intégrateur précédent, on peut déplacer le pôle sur l'axe des réels pour lui donner un module $r$ inférieur à 1, ce qui a pour effet de stabiliser le filtre.




$$H(z) = b_0\frac{z+1}{z-r} \tag{16}$$

Il s'agit d'un filtre passe-bas dont le gain dans la bande passante ($z = 1$) est :

$$G = b_0\frac{2}{1-r} \tag{17}$$

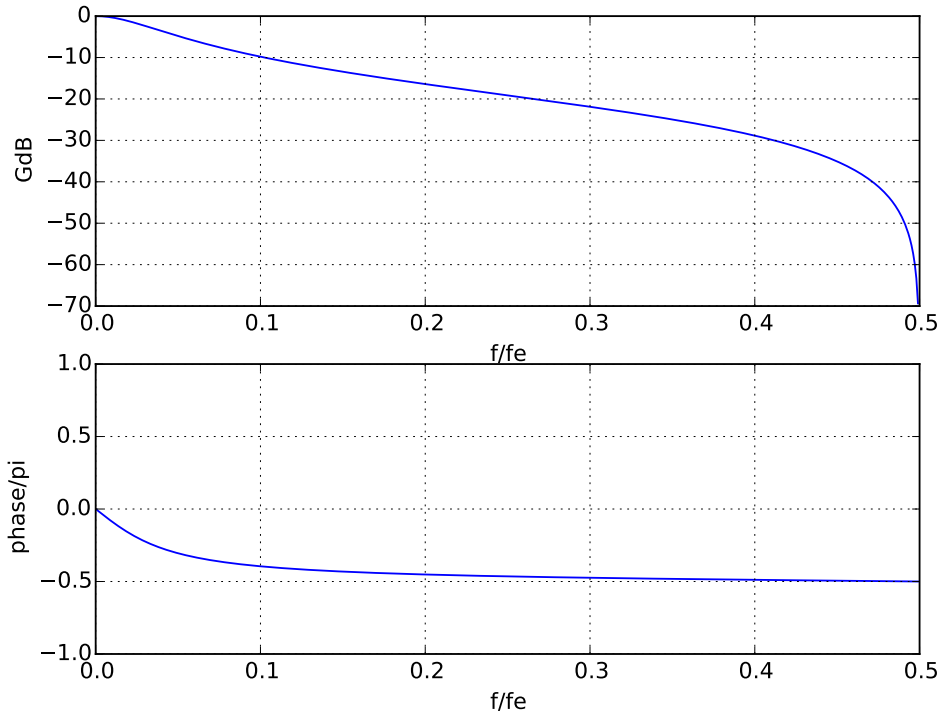
On peut par exemple choisir $b_0$ pour que $G = 1$.

La relation de récurrence est :

$$y_n = ry_{n-1} + b_0(x_n + x_{n-1}) \tag{18}$$

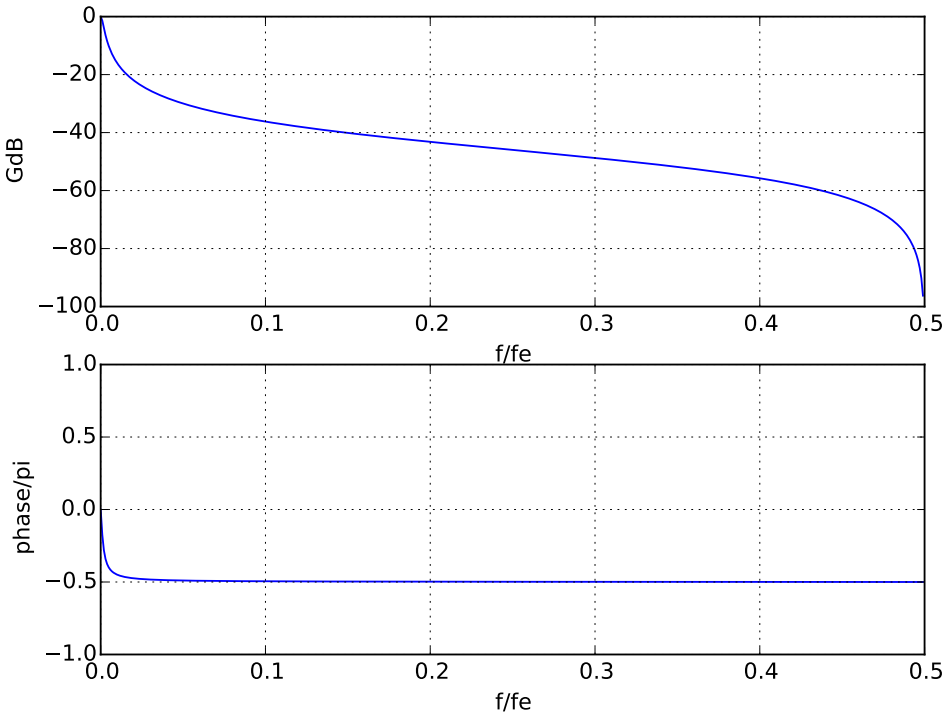
Voici un exemple :

```
r=0.8
G=1
b0 = (1-r)*G/2
b=[b0,b0]
a=[1,-r]
w,h=scipy.signal.freqz(b,a)

figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```



Le filtre obtenu est similaire à une filtre du premier ordre calculé par transformation bilinéaire d'une fonction de transfert analogique. La méthode de placement des pôles et zéros permet d'obtenir facilement un filtre passe-bas ayant une très basse fréquence de coupure, par exemple :

```
r=0.99
G=1
b0 = (1-r)*G/2
b=[b0,b0]
a=[1,-r]
w,h=scipy.signal.freqz(b,a)

figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```




On obtient ainsi un intégrateur stable, dont le gain à fréquence nulle est fini. Pour cette utilisation, il peut être utile d'ajuster la constante $b_0$ pour obtenir une intégration vraie

du signal d'entrée. On doit alors avoir dans la plage de fréquence d'intégration :

$$b_0 \frac{z+1}{z-r} = \frac{T_e}{2} \frac{z+1}{z-1} \tag{19}$$

ce qui donne :

$$b_0 = \frac{T_e}{2} \frac{z-r}{z-1} \tag{20}$$

Si $r$ est très proche de 1, et si $z$ est différent de 1 (fréquence non nulle), la fraction est pratiquement égale à 1. Cela démontre le comportement intégrateur et indique le choix de la constante :

$$b_0 = \frac{T_e}{2} \tag{21}$$

On obtient finalement la relation de récurrence suivante pour une utilisation de ce filtre en tant qu'intégrateur :

$$y_n = r y_{n-1} + \frac{T_e}{2}(x_n + x_{n-1}) \tag{22}$$

La constante $r$ est inférieure à 1 mais proche de 1, d'autant plus que l'on veut une fréquence de coupure basse. Pour $r = 1$, on retrouve l'intégrateur parfait, instable à fréquence nulle.

Pour une utilisation en tant que filtre passe-bas avec un gain unité dans la bande passante, la relation de récurrence est :

$$y_n = r y_{n-1} + \frac{1-r}{2}(x_n + x_{n-1}) \tag{23}$$

On peut chercher à obtenir ce filtre passe-bas directement à partir de l'équation différentielle du système à temps continu. Pour une pulsation analogique de coupure $\omega_c$ et un gain dans la bande passante $g$, cette équation est :

$$\frac{dy(t)}{dt} + \omega_c y(t) = \omega_c g x(t) \tag{24}$$

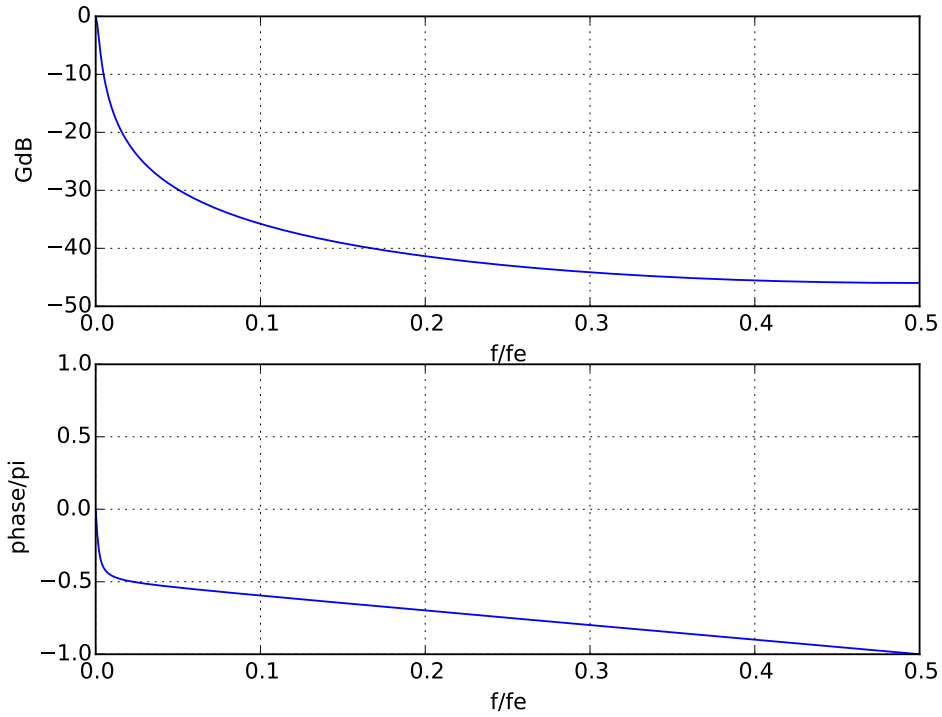
La méthode d'intégration d'Euler s'écrit :

$$\begin{aligned} y_n &= y_{n-1} + \omega_c \frac{T_e}{2}(g x_{n-1} - y_{n-1}) && (25)\\ &= (1 - \omega_c T_e) y_{n-1} + \omega_c T_e g x_{n-1} && (26)\\ &= r y_{n-1} + (1-r) g x_{n-1} && (27) \end{aligned}$$

Cette relation est différente de la relation (23) puisque la méthode d'Euler est une méthode à un pas. La fonction de transfert correspondante a l'inconvénient d'être démunie de zéro. Voyons sa réponse fréquentielle :

```
b=[0,1-r]
a=[1,-r]
w,h=scipy.signal.freqz(b,a)
```

```
figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```




Cette réponse est très différente de la réponse fréquentielle du filtre analogique. En particulier, on ne retrouve pas du tout le comportement intégrateur dans la bande atténuante. La solution consiste à appliquer une méthode numérique à deux pas, du type Adams-Moulton ([3]), comme on le fait pour l'intégrateur (méthode des trapèzes) :

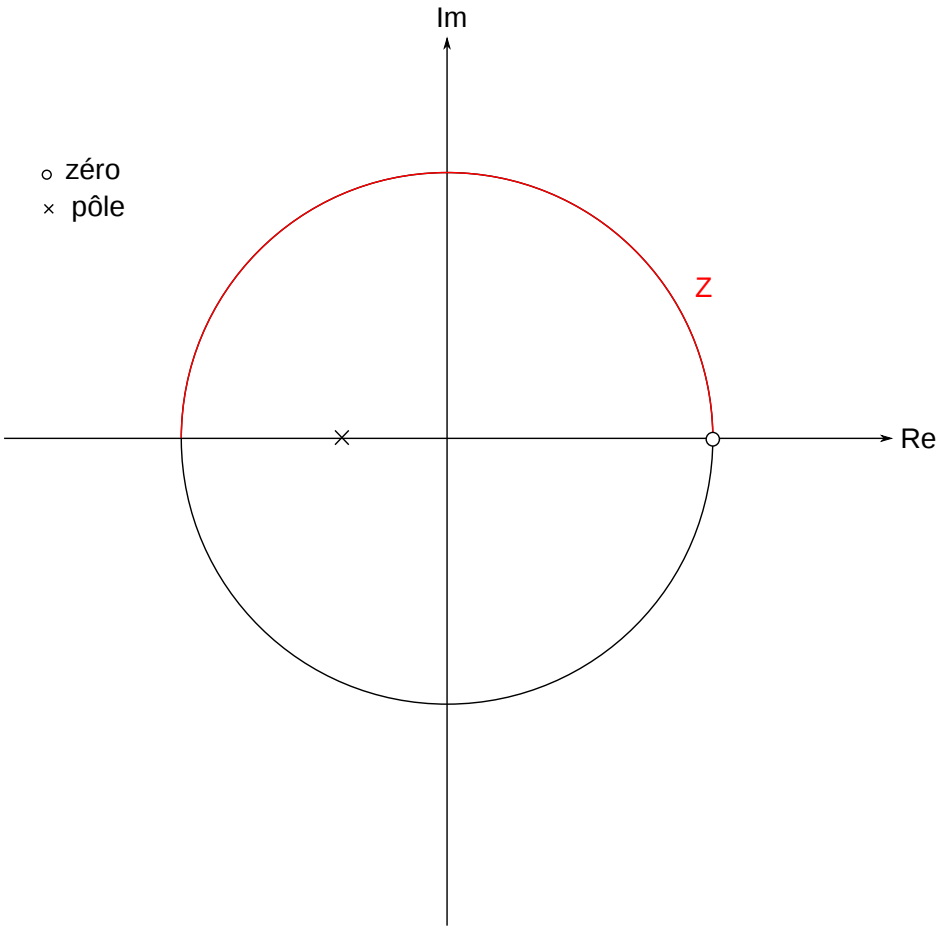
$$y_n = (1 - \omega_c T_e) y_{n-1} + \omega_c T_e \frac{1}{2} g(x_n + x_{n-1}) \tag{28}$$

qui est bien de la forme (23) si l'on pose $r = 1 - \omega_c T_e$ et $g = 1$. La relation réalisant une intégration vraie dans la bande atténuante s'obtient en posant :

$$\omega_c g = 1 \tag{29}$$

### 3.c. Filtre passe-haut

Un filtre passe-haut est obtenu avec un pôle $p_1 = -r$ pour la fréquence de Nyquist et un zéro $q_1 = 1$ pour la fréquence nulle.




$$H(z) = b_0 \frac{z-1}{z+r} \tag{30}$$

La relation de récurrence est :

$$y_n = -r y_{n-1} + b_0 (x_n - x_{n-1}) \tag{31}$$

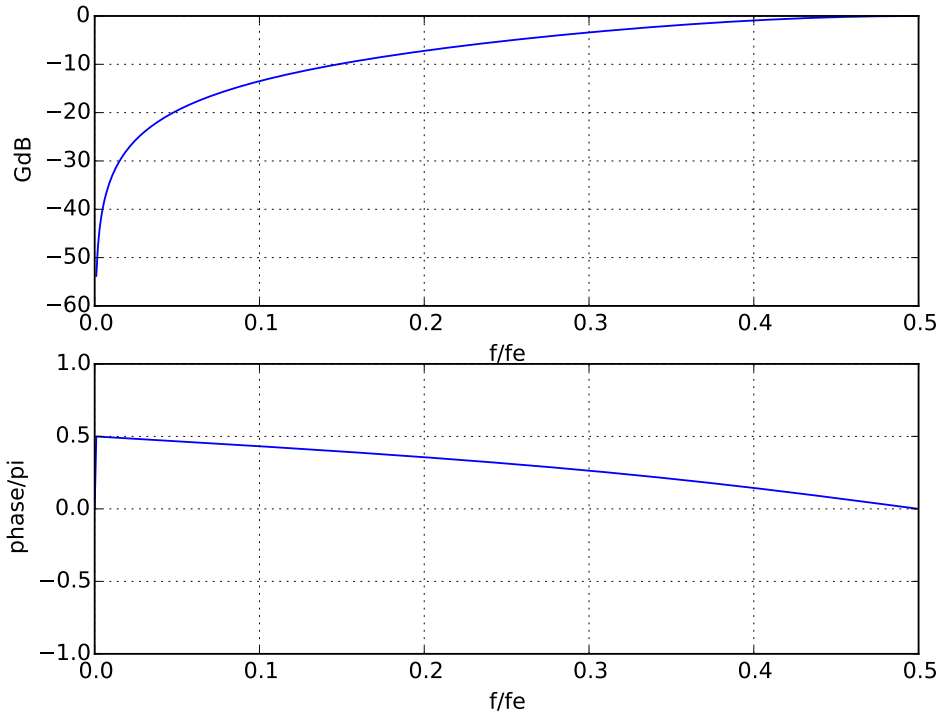
Le gain dans la bande passante ($z = -1$) est :

$$G = b_0 \frac{2}{1-r} \tag{32}$$

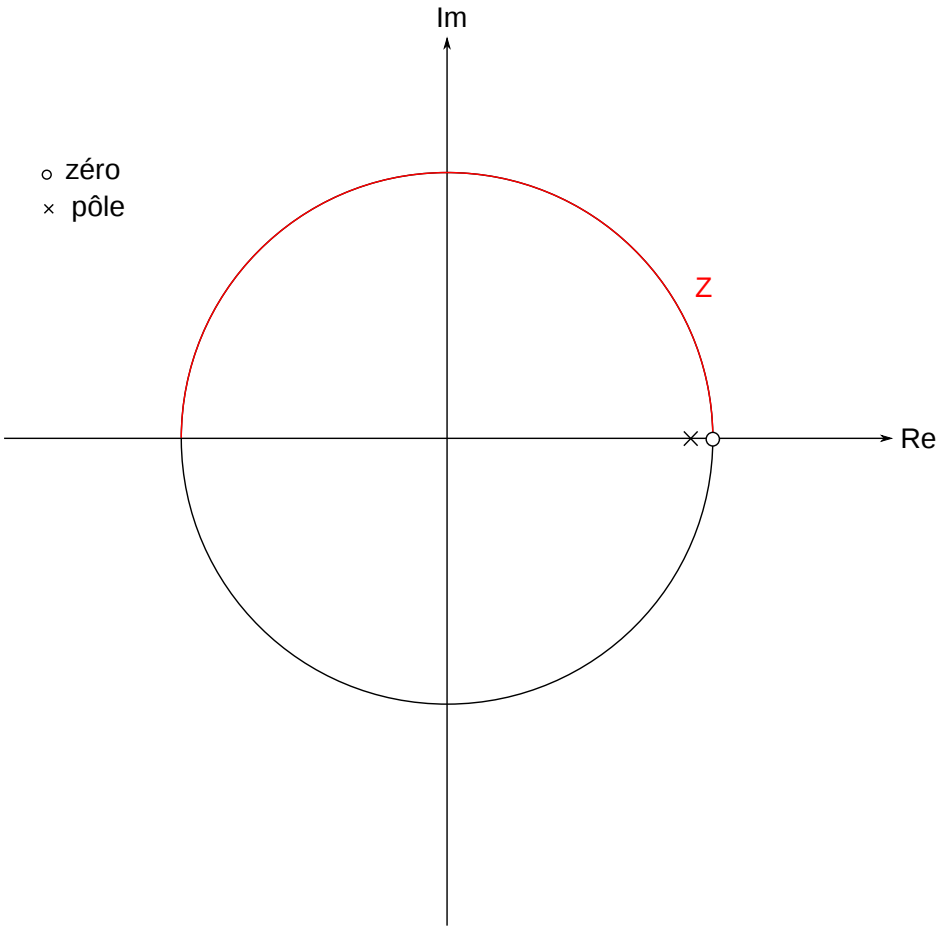
qui est la même relation que pour le filtre passe-bas. Voici un exemple :

```
r=0.2
G=1
b0 = (1-r)*G/2
b=[b0,-b0]
```

```
a=[1,r]
w,h=scipy.signal.freqz(b,a)

figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```




Pour obtenir un filtre bloquant la composante continue, il faut abaisser la fréquence de coupure. Cela se fait en plaçant le pôle en $p_1 = r$, avec $r$ très proche de 1 :




$$H(z) = b_0 \frac{z-1}{z-r} \tag{33}$$

Le gain dans la bande passante est :

$$G = b_0 \frac{2}{1+r} \tag{34}$$

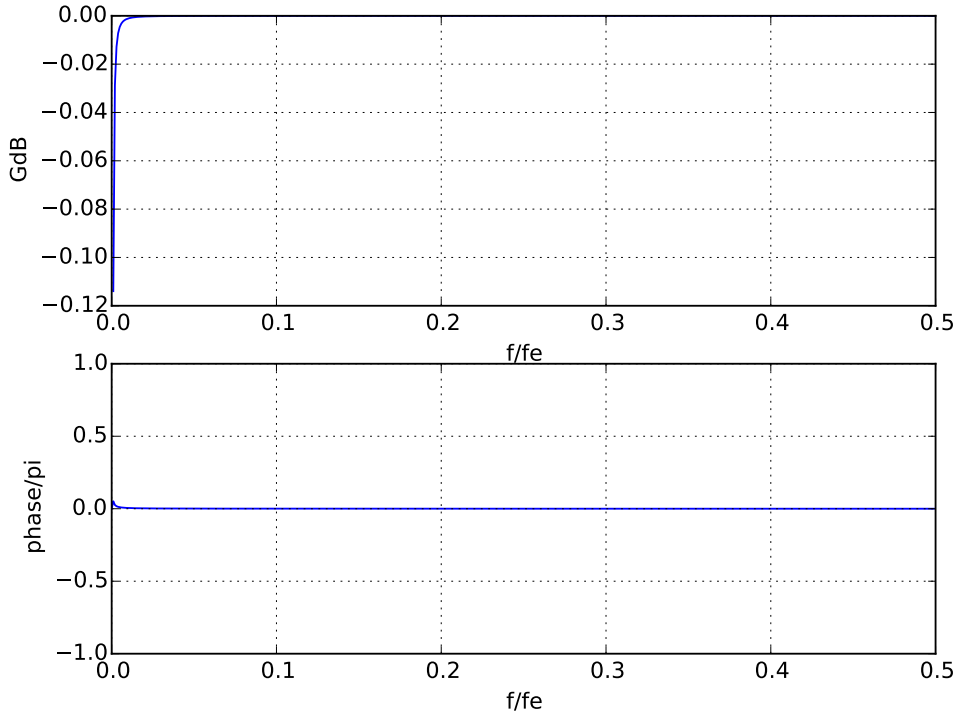
La relation de récurrence est :

$$y_n = r y_{n-1} + b_0 (x_n - x_{n-1}) \tag{35}$$

Voici un exemple. Si est $r$ est très proche de 1, on a pratiquement $b_0 = 1$.

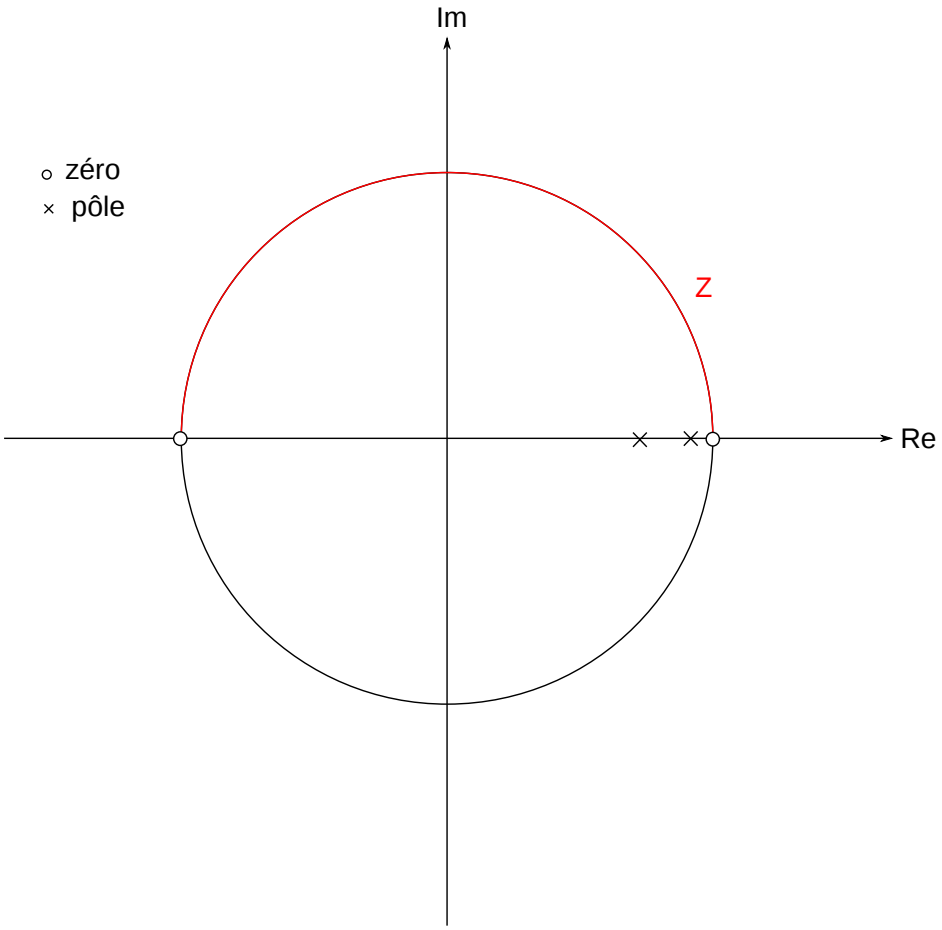
```
r=0.999
G=1
b0 = (1+r)*G/2
b=[b0,-b0]
a=[1,-r]
w,h=scipy.signal.freqz(b,a)

figure()
```

```
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```




### 3.d. Filtre passe-bande

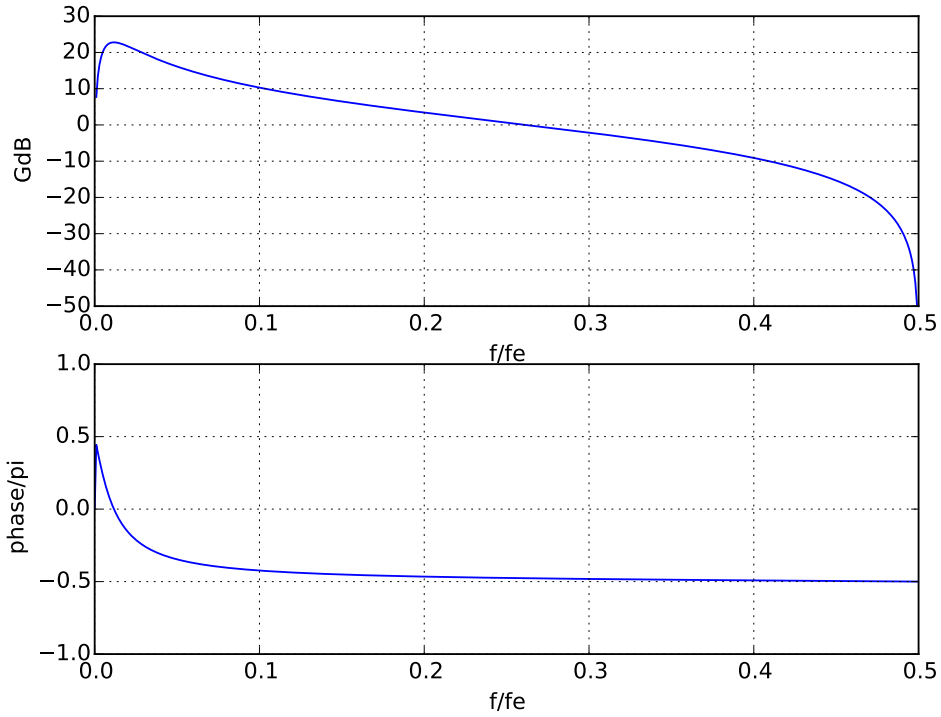
Un filtre passe-bande peut être obtenu en associant en série un filtre passe-bas et un filtre passe-haut :




$$H(z) = b_0 \frac{z-1}{z-r_1}\frac{z+1}{z-r_2} = b_0 \frac{1-z^{-2}}{1-(r_1+r_2)z^{-1}+r_1 r_2 z^{-2}} \tag{36}$$

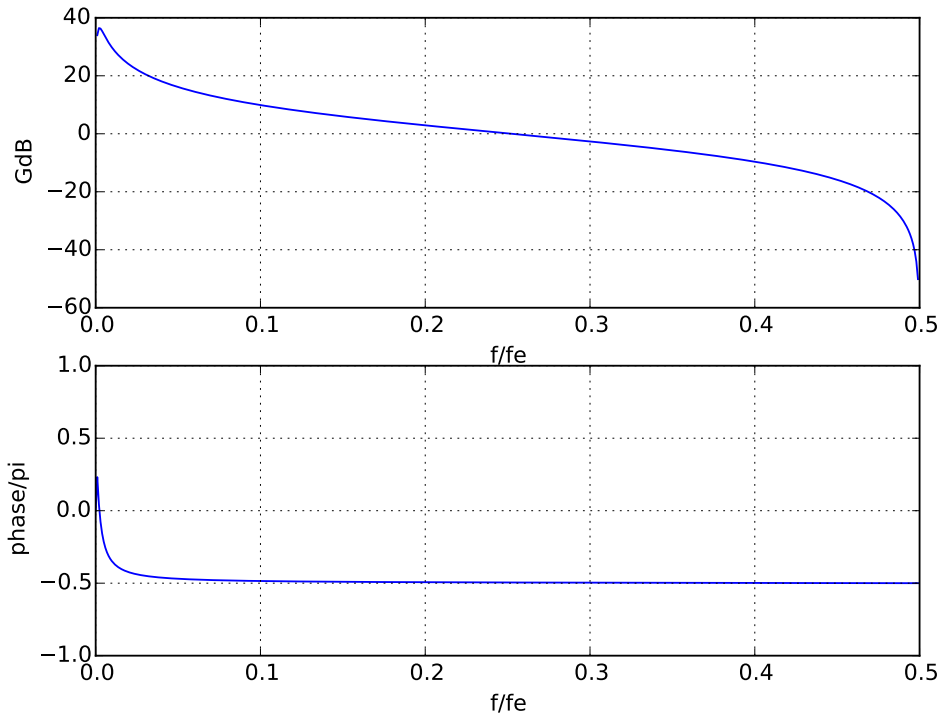
```
r1=0.9
r2=0.95

b=[1,0,-1]
a=[1,-(r1+r2),r1*r2]
w,h=scipy.signal.freqz(b,a)

figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
```

```
axis([0,0.5,-1,1])
grid()
```



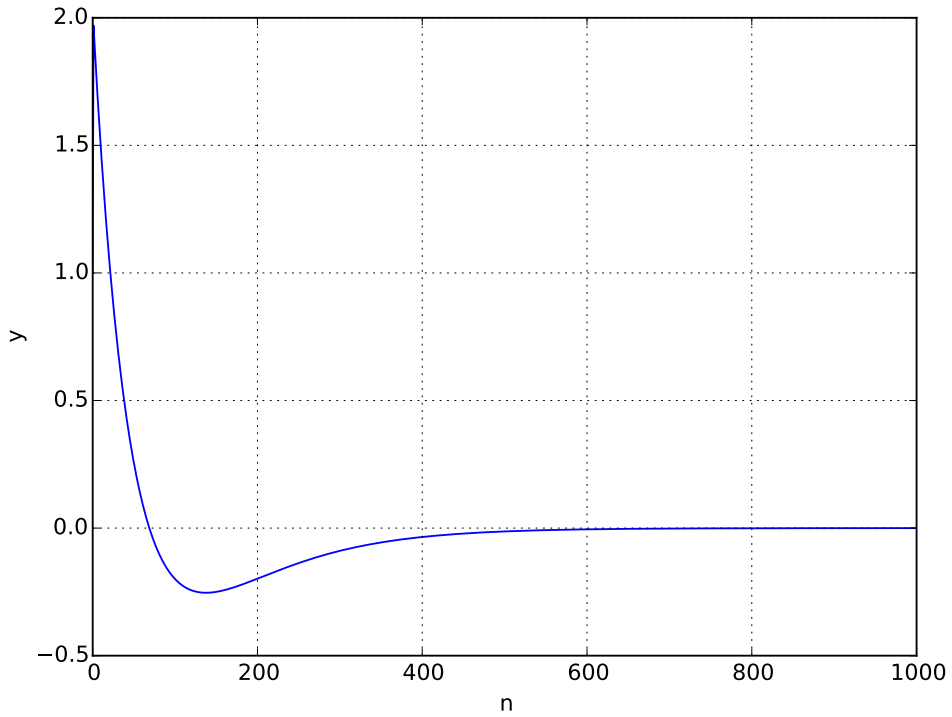

Une application de ce type de filtre est un intégrateur avec un gain nul à fréquence nulle, obtenu avec $r_1$ et $r_2$ très proches de 1.

```
r1=0.98
r2=0.99

b=[1,0,-1]
a=[1,-(r1+r2),r1*r2]
w,h=scipy.signal.freqz(b,a)

figure()
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```




Voyons la réponse impulsionnelle de ce filtre :

```
x = numpy.zeros(1000)
x[0] = 1.0
zi = scipy.signal.lfiltic(b,a,y=[0,0],x=[0,0])
[y,zf] = scipy.signal.lfilter(b,a,x,zi=zi)

figure()
plot(y)
xlabel("n")
ylabel("y")
grid()
```




Il faut environ 600 échantillons pour atteindre le régime stationnaire. Pour obtenir une réponse plus rapide, il faut augmenter la fréquence de la bande passante, en réduisant $r_1$ et $r_2$.

Pour obtenir une intégration vraie dans la bande atténuante, il faut choisir la constante $b_0$ pour que l'égalité suivante soit à peu près vérifiée dans la bande atténuant :

$$b_0 \frac{z-1}{z-r_1} \frac{z+1}{z-r_2} = \frac{T_e}{2} \frac{z+1}{z-1} \tag{37}$$

Lorsque $r_1$ et $r_2$ sont très proches de 1 et lorsque $z$ est différent de 1 (fréquence non nulle), cette égalité est approximativement vérifiée si :

$$b_0 = \frac{T_e}{2} \tag{38}$$

Voici la relation de récurrence de cet intégrateur :

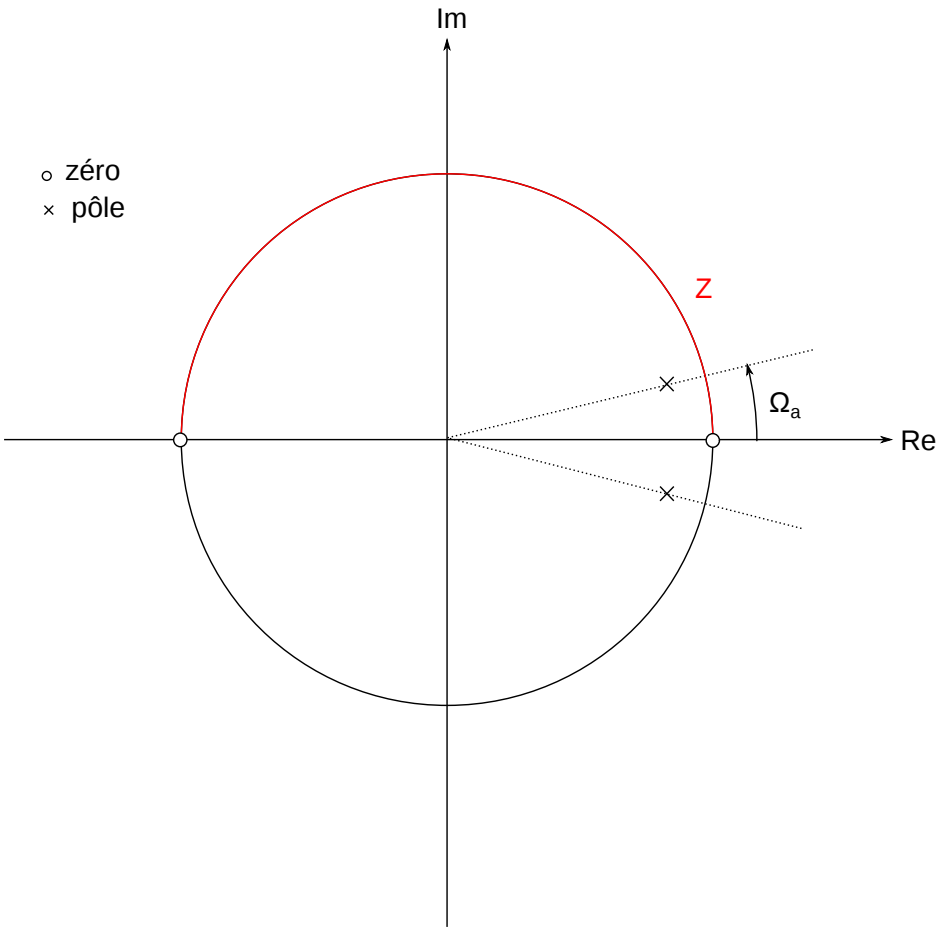
$$y_n = (r_1 + r_2) y_{n-1} - r_1 r_2 y_{n-2} + \frac{T_e}{2}(x_n - x_{n-2}) \tag{39}$$

Si le signal d'entrée comporte une composante de fréquence nulle (composante continue), l'intégration se fait sans décalage en sortie. L'inconvénient est la lenteur de la réponse transitoire. En pratique, il faut ajuster la fréquence de la bande passante juste en dessous de la plus basse fréquence du signal.

# 4. Filtres du second ordre

## 4.a. Filtre passe-bande

Le filtre passe-bande précédent était défini avec deux pôles réels. Pour obtenir un résonateur, on définit deux pôles complexes conjugués à la pulsation de résonance, avec deux zéros $q_1 = 1$ et $q_2 = -1$ qui permettent d'annuler le gain à fréquence nulle et à la fréquence de Nyquist :

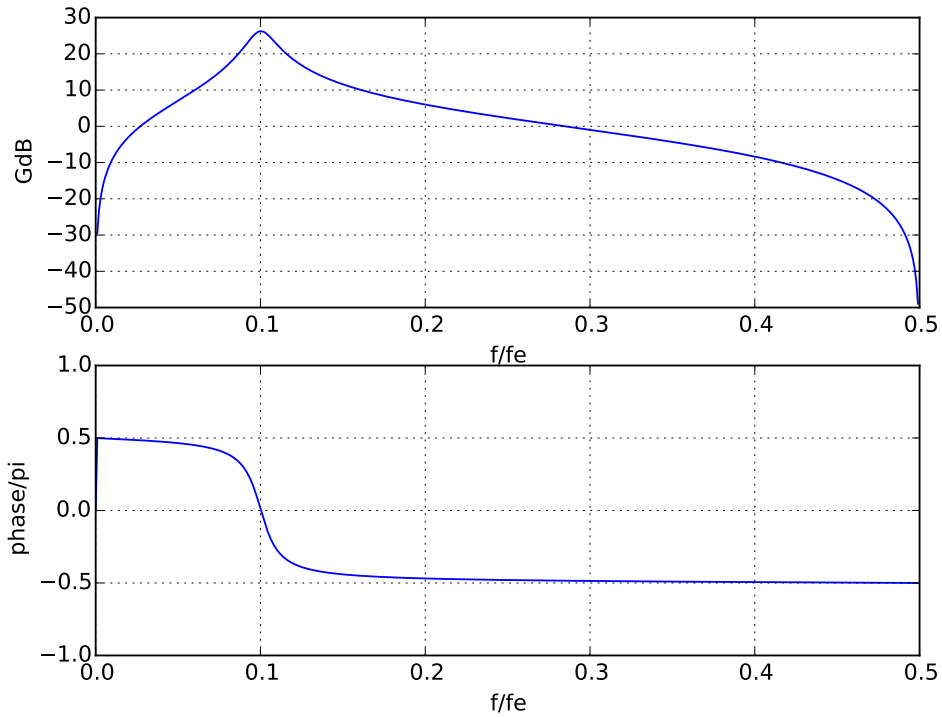



$$H(z) = b_0 \frac{(z-1)(z+1)}{(z - re^{i\Omega_a})(z - re^{-i\Omega_a})} = b_0 \frac{1 - z^{-2}}{1 - 2r\cos(\Omega_a)z^{-1} + r^2 z^{-2}} \tag{40}$$

Voici un exemple, avec une fréquence de résonance égale à 1/5 ième de la fréquence de Nyquist :

```
omega_a=numpy.pi/5
r=0.95
b=[1,0,-1]
a=[1,-2*r*numpy.cos(omega_a),r*r]
w,h=scipy.signal.freqz(b,a)

figure()
```

```
subplot(211)
plot(w/(2*numpy.pi),20*numpy.log10(numpy.absolute(h)))
xlabel("f/fe")
ylabel("GdB")
grid()
subplot(212)
plot(w/(2*numpy.pi),numpy.angle(h)/numpy.pi)
xlabel("f/fe")
ylabel("phase/pi")
axis([0,0.5,-1,1])
grid()
```

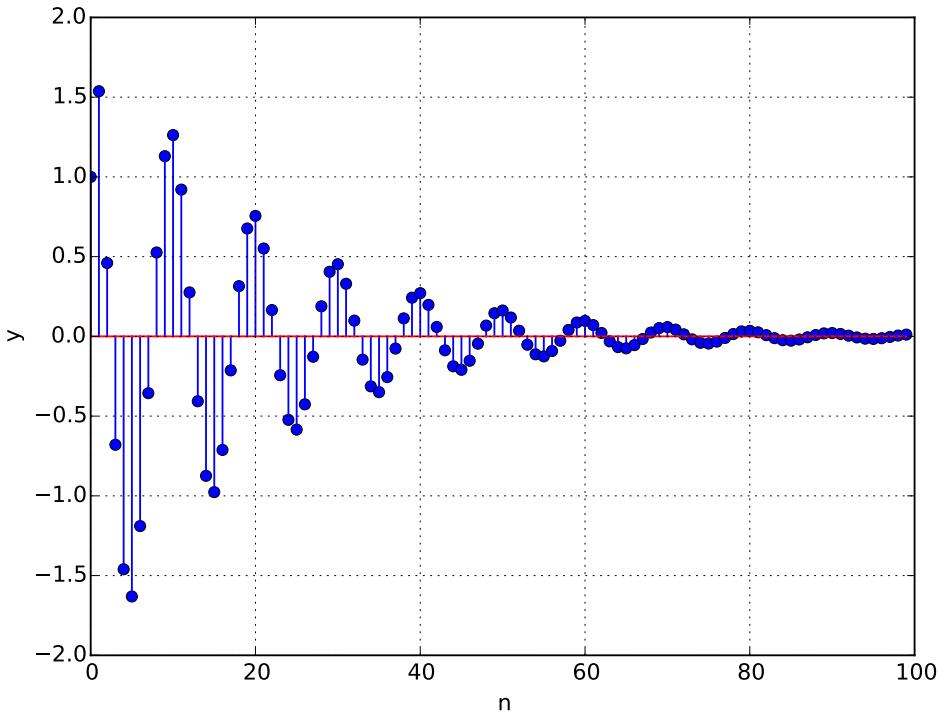



Il faudra bien sûr choisir une constante $b_0$ adéquate pour obtenir un gain unité dans la bande passante.

Voyons la réponse impulsionnelle de ce filtre :

```
x = numpy.zeros(100)
x[0] = 1.0
zi = scipy.signal.lfiltic(b,a,y=[0,0],x=[0,0])
[y,zf] = scipy.signal.lfilter(b,a,x,zi=zi)

figure()
stem(y)
xlabel("n")
ylabel("y")
```

```
grid()
```




Elle présente des oscillations à la fréquence de résonance, égale à un dixième de la fréquence d'échantillonnage. Elle tend bien vers zéro puisque ce filtre est stable (ses pôles ont un module strictement inférieur à 1).

**Références**

[1] J.O. Smith, *Introduction to digital filters, with audio applications*, (BookSurge, 2007)
[2] Tan Li, Jiang Jean, *Digital signal processing : fundamentals and applications*, (Elsevier, 2013)
[3] J.P. Demailly, *Analyse numérique et équations différentielles*, (P.U.G., 1991)