Les fitres à réponse impulsionnelle infinie en durée

(RII)

Objectifs de l'apprentissage

- Introduction à la théorie des filtres RII :
  - Propriétés.
  - Calcul des coefficients.
  - Architectures de mise en oeuvre
- Mise en oeuvre en Matlab et C

Introduction

- Les filtres RIF sont caractérisés par une fonction de transfert exprimée par une fraction rationnelle en z^{-1} :

$$H(z) = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{1 + \sum_{k=1}^{M-1} a_k z^{-k}}$$

- L'introduction de pôles dans l'expression de H(z) permet de réduire considérablement le nombre de coefficients du filtre par rapport à un filtre équivalent RIF.
- Propriétés :
  - Très efficaces en temps de calcul
  - Réponse en phase non lineaire en général
  - Leur stabilité doit être vérifiée lors de la conception

Propriétés d'un filtre RII

- Équation d'e/s :

$$y[n] = \sum_{k=0}^{N-1} b[k]x[n-k] + \sum_{k=1}^{M-1} a[k]y[n-k]$$

x[n] représente les valeurs successives du signal d'entrée,

a_k, b_k représentent les coefficients de la fonction de transfert du filtres,

y[n] représente les valeurs successives du signal de sortie,

N, *M* représentent les ordres du numérateur et du dénominateur de *H(Z)* (*M* est souvent appelé l'ordre du filtre).

Propriétés d'un filtre RII

- H(z) peut être factorisé pour donner :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{M} a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}}$$

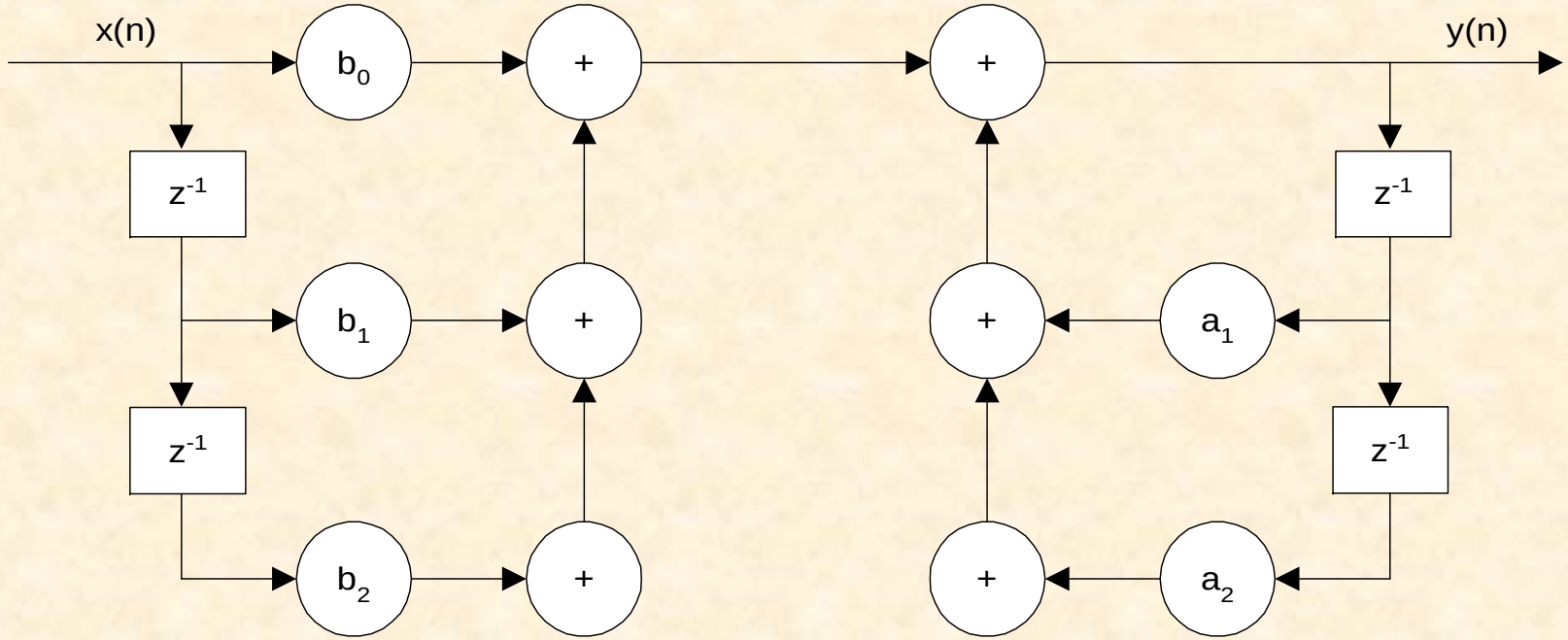
$$= k \frac{(z - z_1)(z - z_2)\cdots(z - z_N)}{(z - p_1)(z - p_2)\cdots(z - p_N)}$$

Où : $z_1, z_2, \ldots, z_N$ sont the zéros,
$p_1, p_2, \ldots, p_N$ sont the pôles.

$\Rightarrow$ Un filtre RII est complètement spécifié par les valeurs des pôles et des zéros de *H(z)* (à *k* près).

Propriétés d'un filtre RII

- Diagramme de flux pour $N=M=2$



- À l'instar des filtres RIF, deux filtres RII se distinguent uniquement par leur coefficients !

$$y[n] = \sum_{k=0}^{N} b[k]x[n-k] + \sum_{k=1}^{M} a[k]y[n-k]$$

Origine du nom RII

- On a : $y[n] = \sum_{k=0}^{N-1} b_k x[n-k] + \sum_{k=1}^{M-1} a_k y[n-k]$
- Si on remplace le signal d'entrée *x[n]* par une impulsion $\delta[n]$, alors :

$$y[n] = h[n] = \sum_{k=0}^{N-1} b_k \delta[n-k] + \sum_{k=1}^{M-1} a_k y[n-k]$$

$$= b_0\delta[n] + b_1\delta[n-1] + \cdots + b_{N-1}\delta[n-(N-1)]$$

$$+ a_1 y[n-1] + a_2 y[n-2] + \cdots + a_{M-1} y[n-(M-1)]$$

Origine du nom RII

- Et puisque :

$$\delta[n-k]=\begin{cases}1 \text{ for } n=k\\ 0 \text{ for } n\neq k\end{cases}$$

On en déduit :

$$h[0]=b_0$$

$$h[1]=b_1-y[0]$$

$$\vdots$$

$$h[n]=\begin{cases}b_n+\sum_{k=1}^{M-1}a_k y[n-k] \;\; si \;\; n\leq N-1\\ \sum_{k=1}^{M-1}a_k y[n-k] \quad autrement\end{cases}$$

- La réponse impulsionnelle dure infiniment !

Observations

- La réponse impulsionnelle dure infiniment !
- Contrairement aux filtres RIF, la réponse du filtre n'est pas simplement égale aux valeurs successives de ses coefficients.
- Pour un filtre d'ordre M :

$$h[n] = \begin{cases} b_n + \sum_{k=1}^{M-1} a_k y[n-k] \;\; si \;\; n \leq N-1 \\ \sum_{k=1}^{M-1} a_k y[n-k] \quad autrement \end{cases}$$

Réponse en fréquence d'un filtre IIR

- La transformée z de

$$y[n] = \sum_{k=0}^{N-1} b_k x[n-k] + \sum_{k=1}^{M-1} a_k y[n-k]$$

est :

$$H(z) = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{1 + \sum_{k=1}^{M-1} a_k z^{-k}}$$

- La fonction de réponse en fréquence du filtre est obtenue en remplaçant z by $e^{j\omega Te}$:

$$H(\omega) = H(z)\Big|_{z=e^{j\omega T_e}} = \frac{\sum_{k=0}^{N-1} b_k e^{-jk\omega T_e}}{1 + \sum_{k=1}^{M-1} a_k e^{-jk\omega T_e}}$$

Réponse en fréquence d'un filtre IIR

- Puisque $e^{-j2\pi k} = 1$, on a :

$$= H\left(\omega + \frac{2\pi}{T_e}\right) = \frac{\sum_{k=0}^{N-1} b_k e^{-jk\left(\omega + \frac{2\pi}{T_e}\right)T_e}}{1 + \sum_{k=1}^{M-1} a_k e^{-jk\left(\omega + \frac{2\pi}{T_e}\right)T_e}} = \frac{\sum_{k=0}^{N-1} b_k e^{-jk\omega T_e}}{1 + \sum_{k=1}^{M-1} a_k e^{-jk\omega T_e}}$$

- Par conséquent : $H\left(\omega + \frac{2k\pi}{T_e}\right) = H(\omega)$

- La réponse en fréquence est périodique avec période $2\pi/T_e$ dans le cercle de rayon unité. Si on normalise T_e à 1, on a $H(\omega + 2k\pi) = H(\omega)$

Stabilité d'un filtre RII

- Il faut s'assurer que le $H(z)$ ne possède pas de pôles à l'intérieur du cercle défini par $z=1$.
- Les pôles situés le long de $z=1$ donnent lieu à des réponses oscillatoires.
- $H(z)$ est souvent exprimé en fonction des puissances négatives de z ; il faut dans ce cas convertir son expression suivant les puissances positives de z avant de déterminer les pôles et leur position dans le plan z.

Réponse en phase d'un filtre RIF

- Partant de :

$$H(\omega) = H(z)\Big|_{z=e^{j\omega T_e}} = \frac{\sum_{k=0}^{N-1} b_k e^{-jk\omega T_e}}{1 + \sum_{k=1}^{M-1} a_k e^{-jk\omega T_e}}$$

- On a :

$$arg\left(H(\omega)\right) = arg\left(\frac{\sum_{k=0}^{N-1} b_k e^{-jk\omega T_e}}{1 + \sum_{k=1}^{M-1} a_k e^{-jk\omega T_e}} \right)$$

$$= arg\left(\sum_{k=0}^{N-1} b_k e^{-jk\omega T_e} \right) - arg\left(1 + \sum_{k=1}^{M-1} a_k e^{-jk\omega T_e} \right)$$

- En général, la réponse en phase n'est pas lineáire !

Comparaison entre RII et RIF

RIF	RII
◆ Stable par défaut ◆ Demande n>> 1 pour une bonne performance ◆ Réponse en phase linéaire si filtre causal ◆ La gamme dynamique des différents états se calcule facilement	◆ La stabilité dépend de la position des pôles de $H(z)$ ◆ Peut donner une performance adéquate pour n=1 ou 2 ◆ Réponse en phase non linéaire en général ◆ La gamme dynamique des différents états se calcule difficilement et peut avoir un impact négatif sur la performance ◆ Effets de quantisation et d'arrondi plus prononcés que pour RIF. ◆ Possibilité de cycles limites

Utilisation pour générer des fonctions

- Le fait que les pôles situés le long de z=1 donnent lieu à des réponses oscillatoires peut servir à générer des fonctions sinusoïdales en temps-réel, avec des fréquences et des amplitudes programmables.

$$\sin(\omega_0.t).u(t) \quad <=> \quad \frac{z.\sin(\omega_0.T_e)}{z^2 - 2.z.\cos(\omega_0.T_e)+1}$$

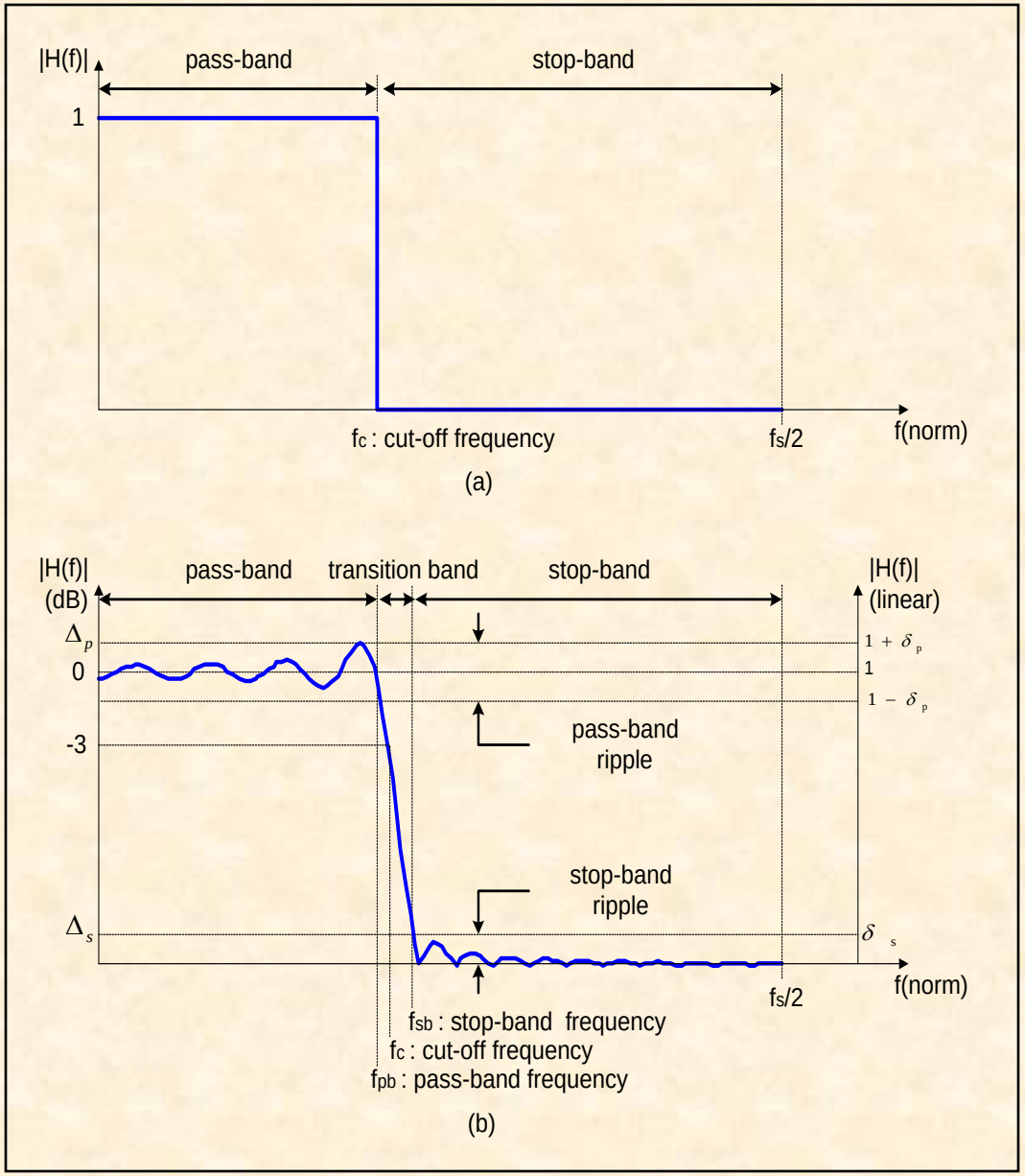
$$\cos(\omega_0.t).u(t) \quad <=> \quad \frac{z.[z - \cos(\omega_0.T_e)]}{z^2 - 2.z.\cos(\omega_0.T_e)+1}$$

- Pour des fonctions périodiques de forme arbitraire, on peut utiliser des tables de valeurs

Conception d’un filtre RII

- Cinq étapes sont requises :
  1. Spécification du filtre
  2. Calcul des coefficients.
  3. Choix d’une architecture de mise en oeuvre.
  4. Simulation (option).
  5. Implémentation.

Étape 1 : spécification du filtre



Étape 2 : calcul des coefficients

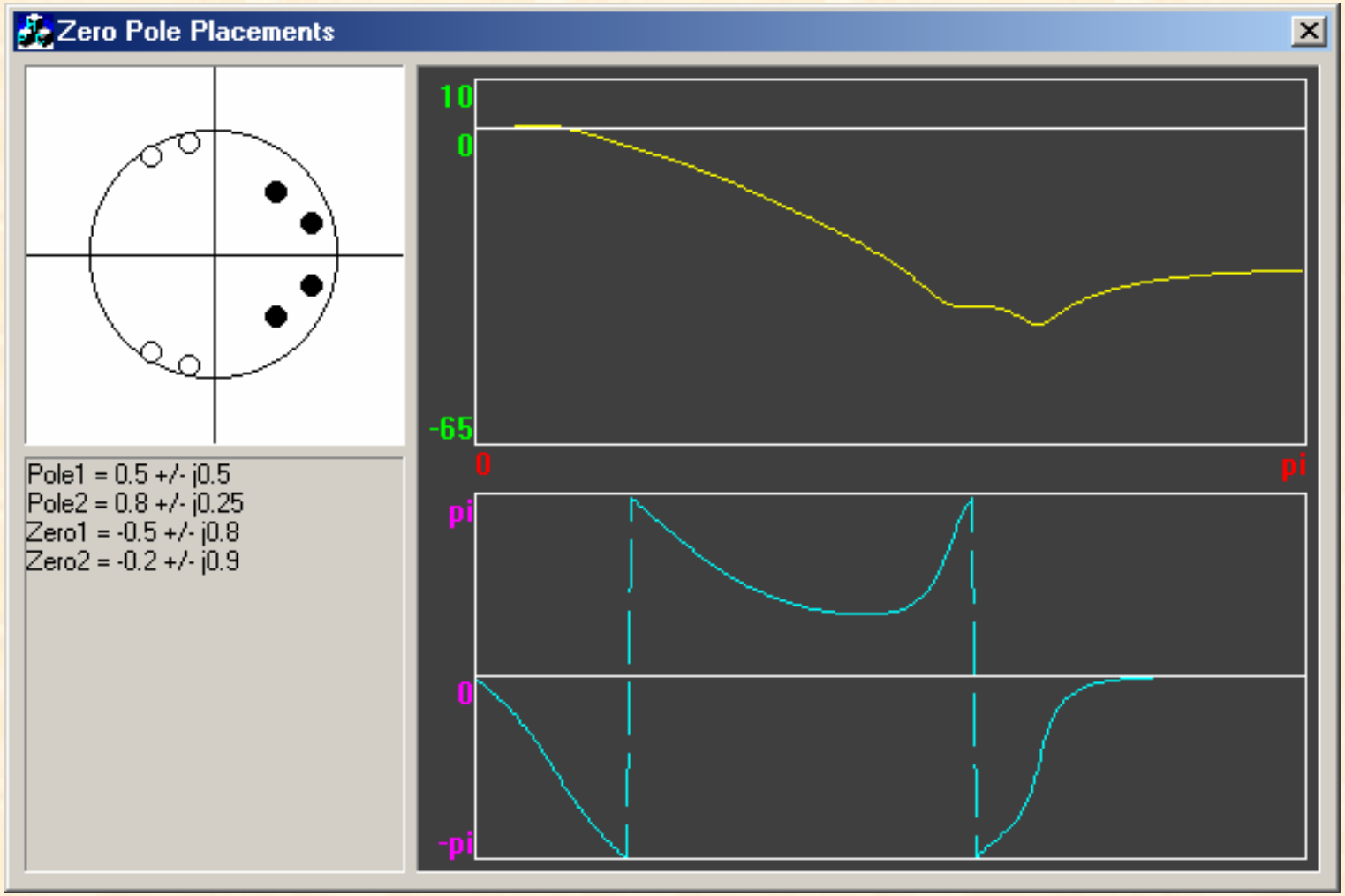
- Il existe deux approches :
  - Placement direct des pôles et zeros dans la plan z
  - Conversion d'un filtre analogique :
    - En utilisant la transformation bilinéaire
    - En utilisant le principe de l'invariance de la réponse impulsionnelle
- Dans le cas de la transformation d'un filtre analogique, on part souvent de l'équation d'un filtre passe bas normalisé que l'on adapte au type désiré (passe haut, passe bande, etc.) avant la conversion.

Méthode du placement des pôles et zéros

- Basée sur le principe que, dans le plan z :
  - Le placement d'un zéros près ou sur le cercle unité dans le plan z minimise la fonction de réponse en fréquence du filtre à cet endroit.
  - Le placement d'un pôle près ou sur le cercle unité dans le plan z maximise la fonction de réponse en fréquence du filtre à cet endroit.
  - Pour obtenir un filtre avec des coefficients réels (donc réalisable), il faut que les pôles et zéros soient à valeurs réelles ou qu'ils apparaissent pas paires conjuguées.
- Méthode intuitive au début mais qui demande beaucoup de calculs ensuite.

Méthode du placement des pôles et zéros

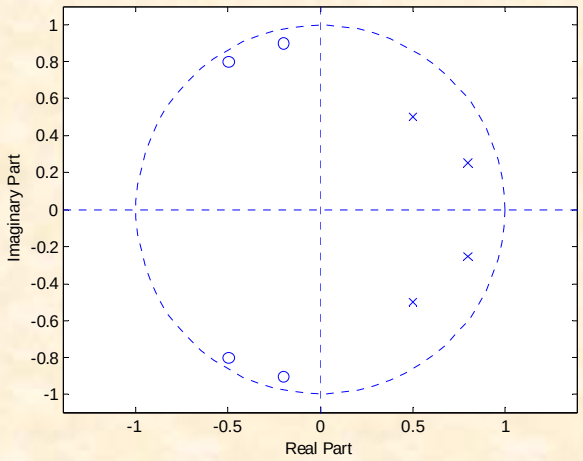
- Exemple :

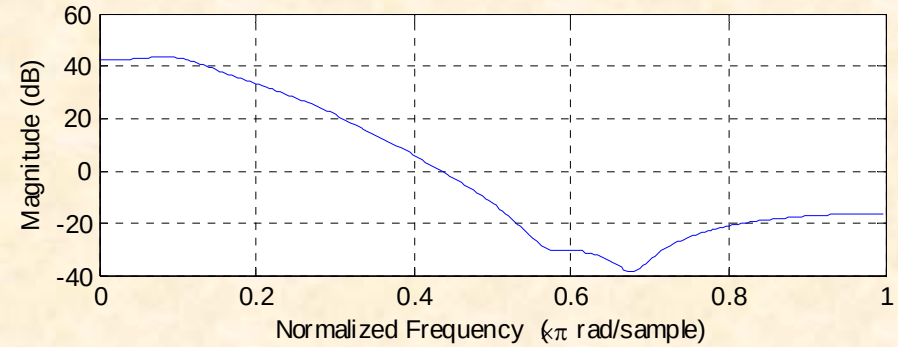


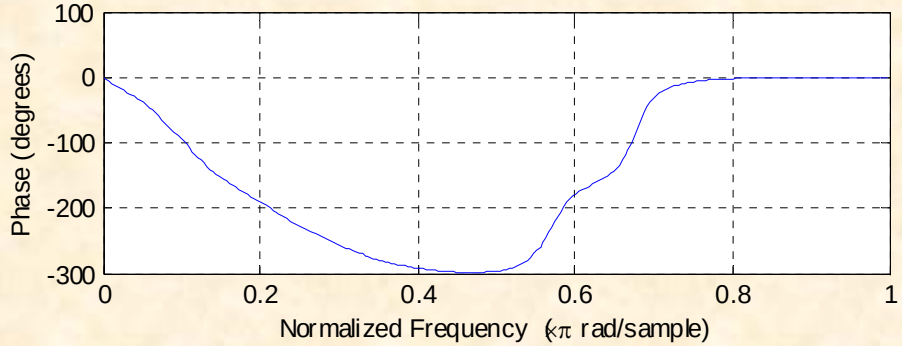
Méthode du placement des pôles et zéros

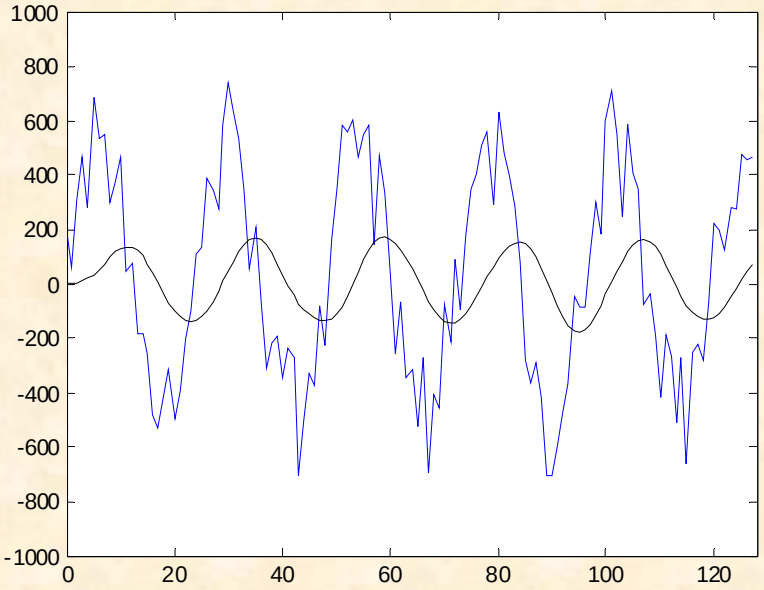
```
% Conception et simulation d'un filtre par placement de pôles et zéros

pole1 = 0.5+0.5i;                    % création de deux paires de pôles conjugués
pole2 = 0.8 +0.25i;
pole3 = conj(pole1); pole4 = conj(pole2);
poles = [pole1 pole2 pole3 pole4];

zero1 = -0.5 + 0.8i;                 % création de deux paires de zéros conjugués
zero2 = -0.2 + 0.9i;
zero3 = conj(zero1); zero4 = conj(zero2);
zeros = [zero1 zero2 zero3 zero4];

denz=poly(poles);                    % conversion des pôles en dénominateur de H(z)
numz=poly(zeros);                    % numérateur de H(z) = 1
zplane(numz, denz);                  % affichage des pôles et zéros
figure(2); freqz(numz,denz,256); % affichage de la réponse en fréquence

t=[0:1:127];                         % test avec 128 valeurs d'un sinus corrompu
x=sin(2*pi*t/24);
x=x+rand(1,128)-0.5;
y=filter(numz,denz,x);
figure(3); plot(t,500*x,'b',t,y,'k');axis([0 128 -1000 1000]);axis('normal');
```

Méthode du placement des pôles et zéros





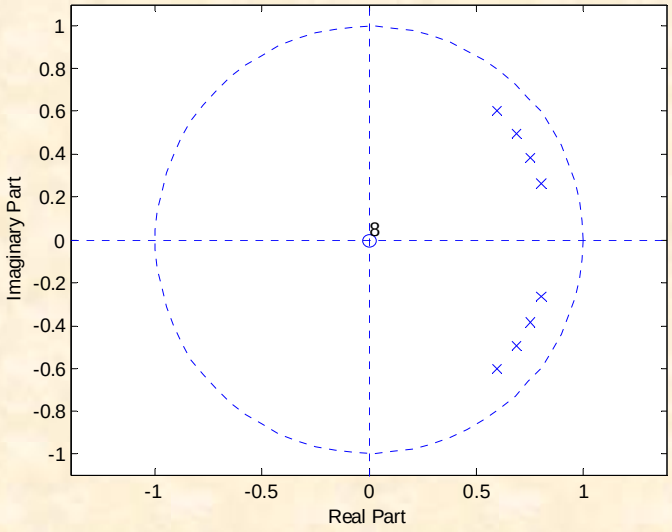


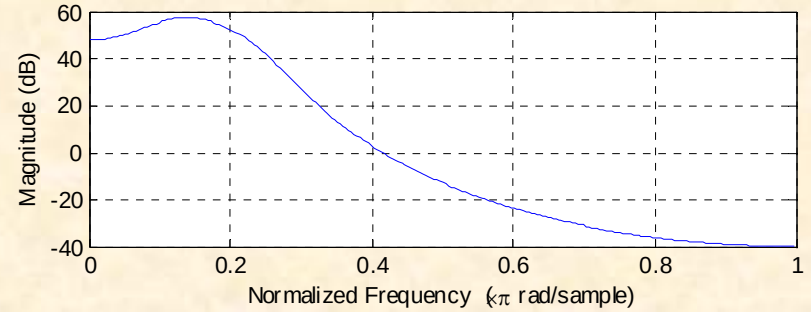


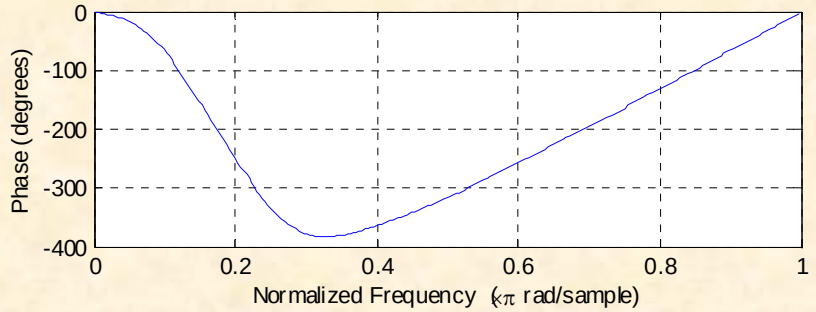
Méthode du placement des pôles et zéros

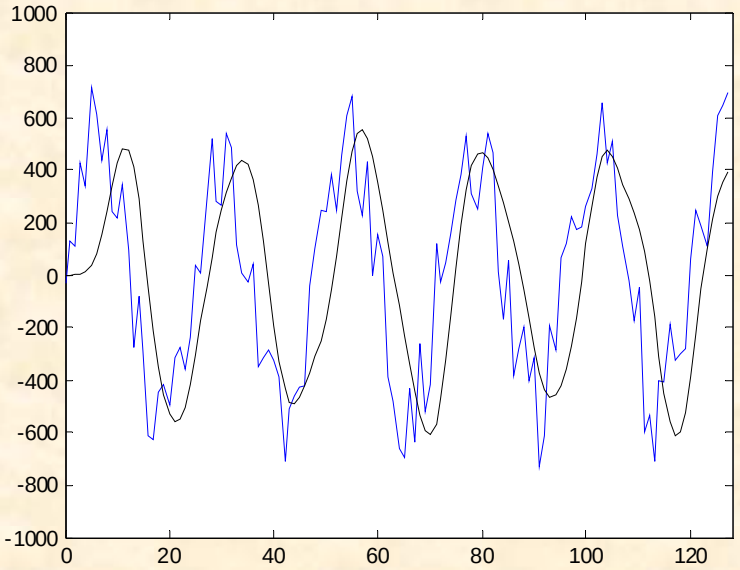
```
% Autre exemple en utilisant des coordonnées polaires

angl=[0.2: 0.1: 0.5]*pi/2;                  % création de 4 paires conjuguées de pôles
poles=0.85*exp(j*angl);
poles=[poles 0.85*exp(-j*angl)];

denz=poly(poles)                            % conversion des pôles en dénominateur de H(z)
numz=[1];                                   % numérateur de H(z) = 1
zplane(numz, denz);                         % affichage des pôles et zéros
figure(2); freqz(numz,denz,256);            % affichage de la réponse en fréquence

t=[0:1:127];                                % test avec 128 valeurs d'un sinus bruité
x=sin(2*pi*t/24);
x=x+rand(1,128)-0.5;
y=filter(numz,denz,x);
figure(3); plot(t,500*x,'b',t,y,'k');axis([0 128 -1000 1000]);axis('normal');
```

Méthode du placement des pôles et zéros









Conversion d'un filtre analogique

- Méthode la plus « simple »
- Exploite le fait qu'il existe des méthodes établies de conception de filtres analogiques.
- Consiste à concevoir un filtre analogique et à le convertir en filtre numérique.
- Les deux méthodes les plus utilisés sont :
  - L'invariance de la réponse impulsionnelle :

$$h[n] = h(t)\Big|_{t=nT_e}$$

  - La transformation bilinéaire :

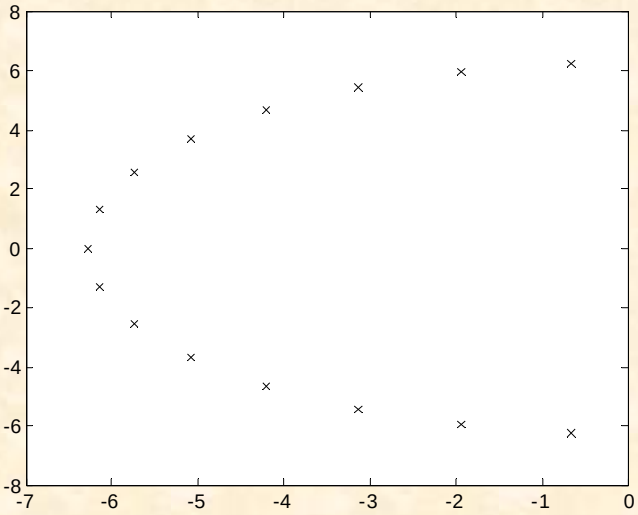
$$H(z) = H(s)\Big|_{s=\ldots}$$

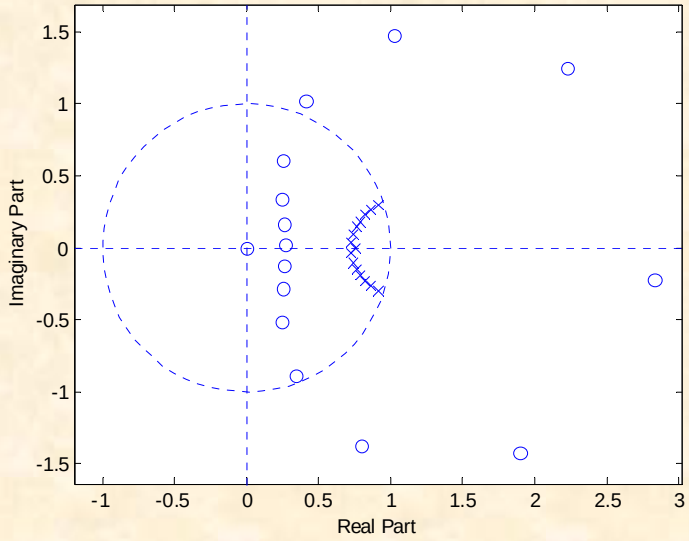
Méthode de l'invariance de la réponse impulsionnelle

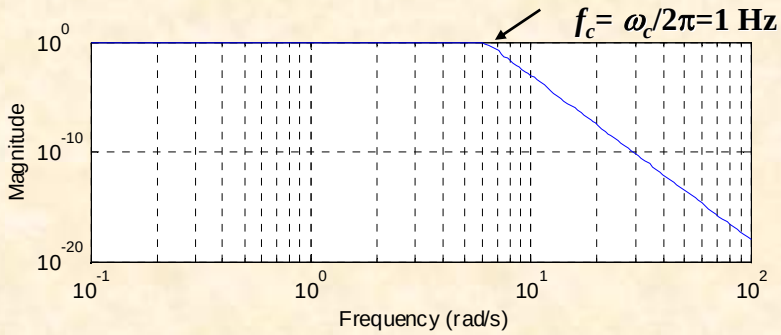
$$h[n] = h(t)\Big|_{t=nT_e}$$

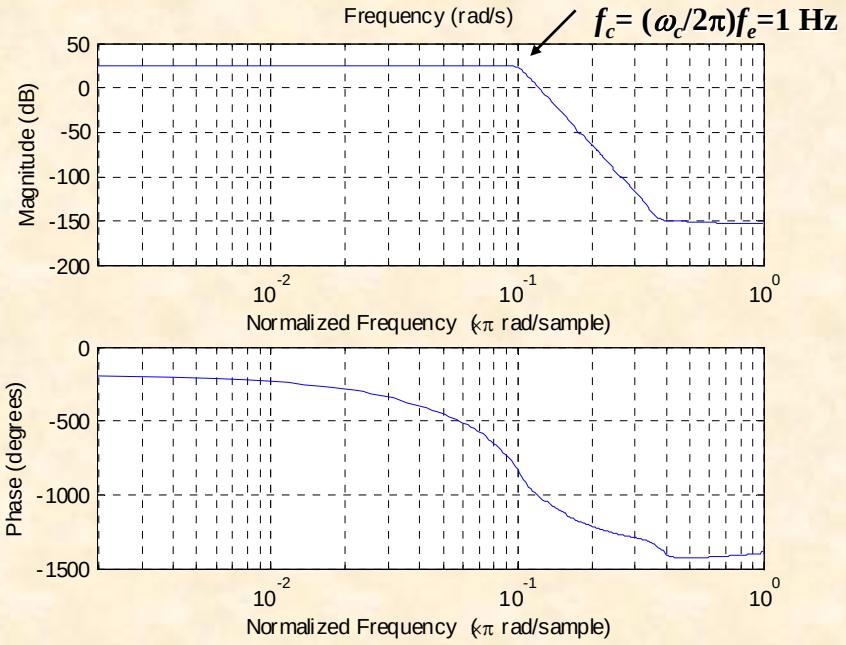
```
% Conception d'un filtre par la méthode de l'invariance de la réponse impulsionnelle

[num,den]=butter(15,2*pi,'s');   % Filtre Passe Bas de Butterworth dans le domaine s avec N=15, fc=1 Hz
[a,p,K] = residue(num,den);      % Décomposition en fractions élémentaires par la méthode des résidus : H(s) = K Σ_i a_i/(s - p_i)
figure(1); plot(p,'xk');         % digramme des pôles maqués par des x noirs
figure(2); freqs(num,den);       % réponse en fréquence analogique

%  A/(s-p_1) + B/(s-p_2)  dans s donne  Ae^{p_1 t} + Be^{p_2 t}  dans t, ce qui donne dans z :
%
%   A z/(z-e^{p_1 T_e}) + B z/(z-e^{p_2 T_e}) = (A+B) + (-Ae^{p_1 T_e})/(z-e^{p_1 T_e}) + (-Be^{p_2 T_e})/(z-e^{p_2 T_e})
%

Te=0.05;                         % fe=20Hz (=> fc normalisé = 1Hz/(fe/2) =0.1)
pz=exp(p*Te);                    % conversion des pôles dans s en des pôles dans z
az=-a.*pz;                       % détermination des coefficients des fractions élémentaires correspondantes
K=K*sum(a);                      % Terme continu
[numz,denz] = residue(az,pz,K);  % détermination de H(z) à partir des fractions élementaires dans z

figure(3); zplane(numz, denz);   % digramme des pôles et zéros dans le plan z
figure(4); freqz(numz,denz);     % réponse en fréquence numérique
```

Méthode de l'invariance de la réponse impulsionnelle









Méthode de la transformation bilinéaire

- Normalement

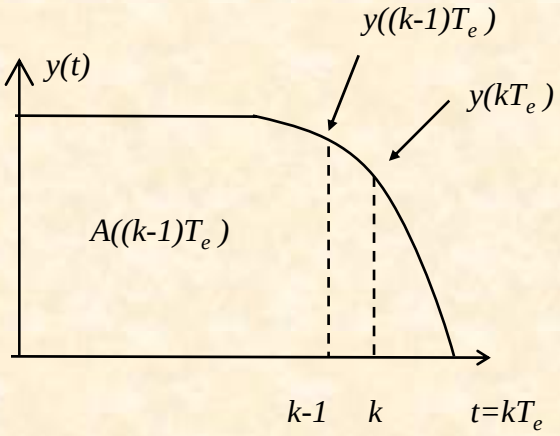
$$z = e^{sT_e} \Rightarrow s = \frac{1}{T_e} ln(z)$$

- On peut donc dériver H(z) de H(s) par la transformation

$$H(z) = H(s)\Big|_{s=\frac{1}{T_e} ln(z)}$$

- Cette transformation donne des équations compliquées
- La transformation bilinéaire utilise l'approximation d'une surface continue par un ensemble de surfaces trapézoïdales.

Méthode de la transformation bilinéaire



$$A(t) = \int_{-\infty}^{t} y(\tau)\,d\tau \qquad \Rightarrow \qquad A(s) = \frac{1}{s} Y(s)$$

$$A(kT_e) = A((k-1)T_e) + \frac{(y(kT_e) + y((k-1)T_e))T_e}{2}$$

$$\Rightarrow A(z) = z^{-1}A(z) + \frac{(1+z^{-1})T_e}{2} Y(z)$$

$$\Rightarrow A(z) = \frac{T_e}{2}\frac{(1+z^{-1})}{(1-z^{-1})} Y(z)$$

- Si $A(t) = A(kT_e)$, alors $\frac{1}{s}$ et $\frac{T_e}{2}\frac{1+z^{-1}}{1-z^{-1}}$ jouent le même rôle dans les domaines s et z
- On peut donc dériver *H(z)* de *H(s)* par la transformation

$$H(z) = H(s)\Big|_{s=\frac{2}{T_e}\frac{1-z^{-1}}{1+z^{-1}}}$$

Méthode de la transformation bilinéaire

- Conséquences dans le domaine jω :

$$s = \frac{2}{T_e}\frac{1-z^{-1}}{1+z^{-1}} \Rightarrow j\omega_a = \frac{2}{T_e}\frac{1-e^{-j\omega_d T_e}}{1+e^{-j\omega_d T_e}} = j\frac{2}{T_e}tg\left(\frac{\omega_d T_e}{2}\right)$$

- Les fréquences dans le domaine ω sont reliées à celles du domaine z par la relation :

$$\omega_a = \frac{2}{T_e}tg\left(\frac{\omega_d T_e}{2}\right)$$

Méthode de la transformation bilinéaire

- Étapes de conception d'un filtre RII par la méthode de la transformation bilinéaire :

  1. Trouver ω_a du filtre analogique à partir de ω_d du filtre numérique : $\omega_a = \frac{2}{T_e} tg\left(\frac{\omega_d T_e}{2}\right)$
  2. Trouver le *H(s)* normalisé du filtre analogique et opérer la transformation *s -> s/* ω_a
  3. Appliquer : $H(z) = H(s)\Big|_{s=\frac{2}{T_e}\frac{1-z^{-1}}{1+z^{-1}}}$

Méthode de la transformation bilinéaire

- On peut accélérer le processus :

$$\omega_a = \frac{2}{T_e} tg\left(\frac{\omega_d T_e}{2}\right) et\, s = \frac{2}{T_e}\frac{1-z^{-1}}{1+z^{-1}} \Rightarrow \frac{\omega_a}{tg\left(\frac{\omega_d T_e}{2}\right)} = \frac{s}{\frac{1-z^{-1}}{1+z^{-1}}}$$

$$\Rightarrow \frac{s}{\omega_a} = \frac{1}{tg\left(\frac{\omega_d T_e}{2}\right)}\frac{1-z^{-1}}{1+z^{-1}}$$

- On peut donc dériver directement H(z) de H(z) par la transformation :

$$H(z) = H(s)\Big|_{s=\frac{1}{tg\left(\frac{\omega_d T_e}{2}\right)}\frac{1-z^{-1}}{1+z^{-1}}}$$

Méthode de la transformation bilinéaire

- Exemple d'utilisation :
  - On veut concevoir un filtre numérique passe bas avec les caractéristiques suivantes :
    - Fréquence de coupure de 6 kHz
    - Fréquence d'échantillonnage de 20 kHz
  - On veut réaliser le filtre à partir d'un filtre analogique équivalent de Butterworth.
  - La fonction de transfert normalisée d'un filtre passe bas de Butterworth de second ordre est :

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}$$

Méthode de la transformation bilinéaire

- Exemple d'utilisation :
  - La fonction de transfert normalisée d'un filtre passe bas de Butterworth de second ordre est :

$$H(s)=\frac{1}{s^2+\sqrt{2}s+1}$$

  - La fonction de transfert dénormalisée dans le domaine z est :

$$H(z)=\left.\frac{1}{s^2+\sqrt{2}s+1}\right|_{s=\frac{1}{a}\frac{1-z^{-1}}{1+z^{-1}}} \quad avec\ a=tg\left(\frac{\omega_a T_e}{2}\right)$$

ou encore :

$$H(z)=\frac{1+2z^{-1}+z^{-2}}{\left(\frac{1+\sqrt{2}a+a^2}{a^2}\right)+2\left(\frac{a^2-1}{a^2}\right)z^{-1}+\left(\frac{1-\sqrt{2}a+a^2}{a^2}\right)z^{-2}}=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}$$

Méthode de la transformation bilinéaire

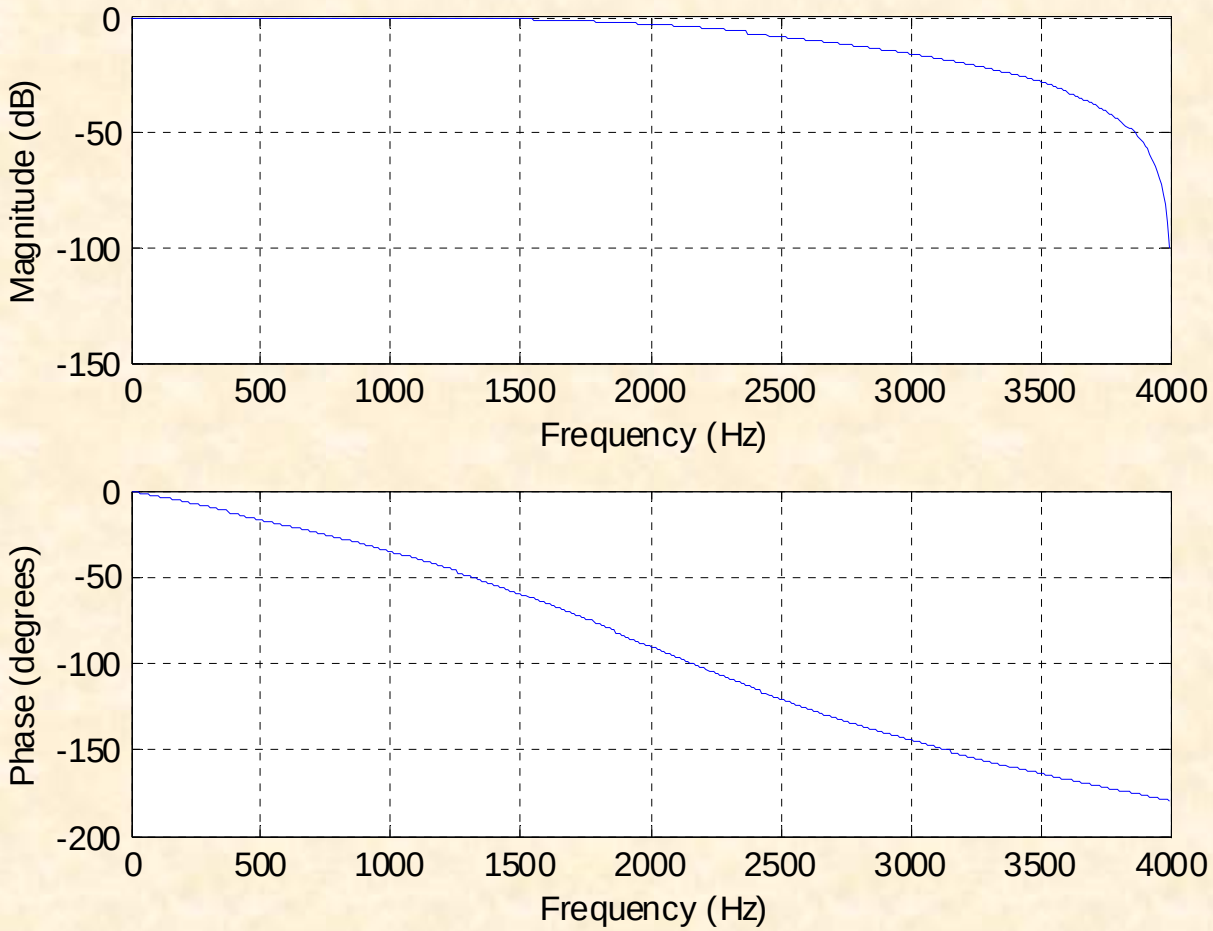
- Exemple de code Matlab pour calculer les coefficients :

```
a   = tan(pi*2/8);          % fréquence de coupure à 2kHz, fe à 8 kHz
b   = 1 + 2^0.5*a + a^2;
b0 = a^2/b;                 %  H(z) = (1 + 2z^-1 + z^-2) / ((1 + sqrt(2)a + a^2)/a^2) + 2((a^2 - 1)/a^2)z^-1 + ((1 - sqrt(2)a + a^2)/a^2)z^-2) = (b0 + b1z^-1 + b2z^-2) / (1 + a1z^-1 + a2z^-2)
b1 = 2*b0;
b2 = b0;
a1 = 2*(a^2 -1)/b;
a2 = (1 - 2^0.5*a + a^2)/b;

numz  = [b0 b1 b2];
denz  = [1 a1 a2];

figure(1)
freqz(numz,denz,512,8000)

fid = fopen('IIR_coef_float.txt', 'w');    % sauvegarde des coefficients
      fprintf(fid,'%0.4f,%0.4f,%0.4f\n',numz);
      fprintf(fid,'%0.4f,%0.4f\n',denz);
fclose(fid);
```

Résultats : numz = [0.1311, 0.2622, 0.1311]

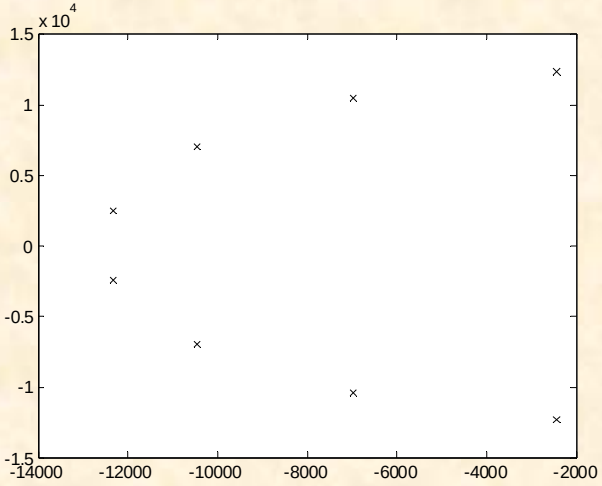
denz = [1, -0.7478, 0.2722]

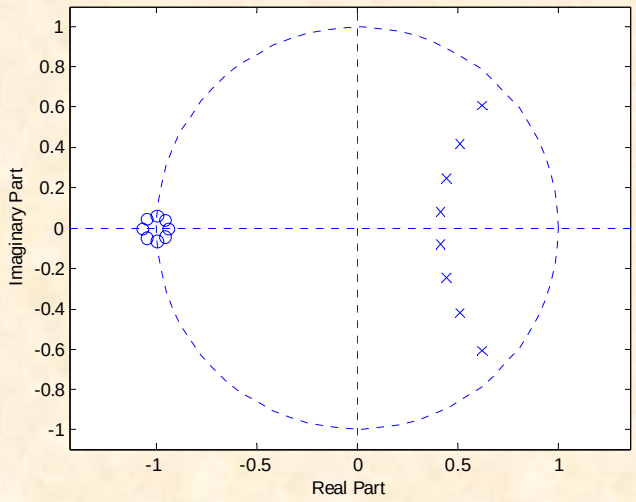
Méthode de la transformation bilinéaire



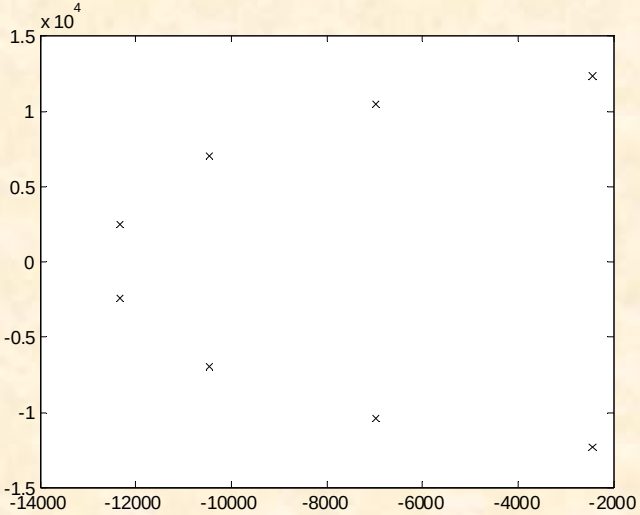
Méthode de la transformation bilinéaire

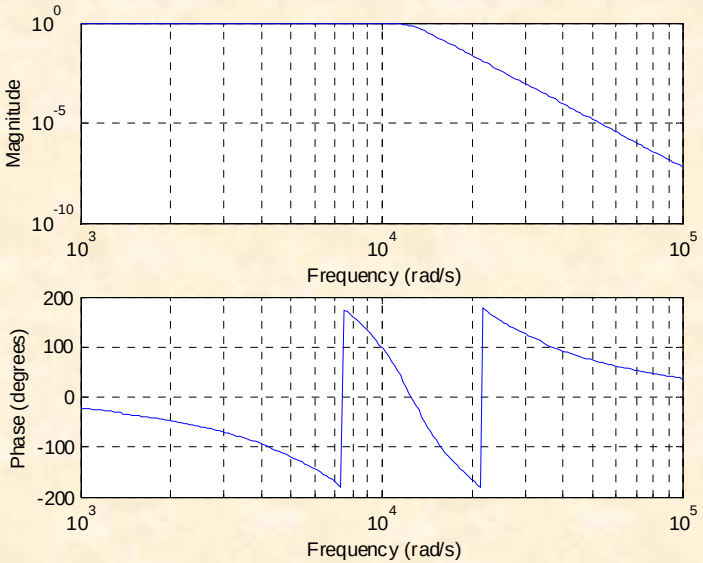
```
[num,den]=butter(8,2*pi*2000,'s');        % Filtre PB de Butterworth dans le domaine s
                                          % avec N=8, fc=2 kHz
figure(1); plot(p,'xk');                  % digramme des pôles, marqués par des x noirs
figure(2); freqs(num,den);                % réponse en fréquence analogique

[numz,denz] = bilinear(num,den,16000);    % transformation bilinéaire avec fe=16 KHz
figure(3); zplane(numz, denz);            % digramme des pôles et zéros dans le plan z
figure(4); freqz(numz,denz,512,16000);    % réponse en fréquence numérique
```

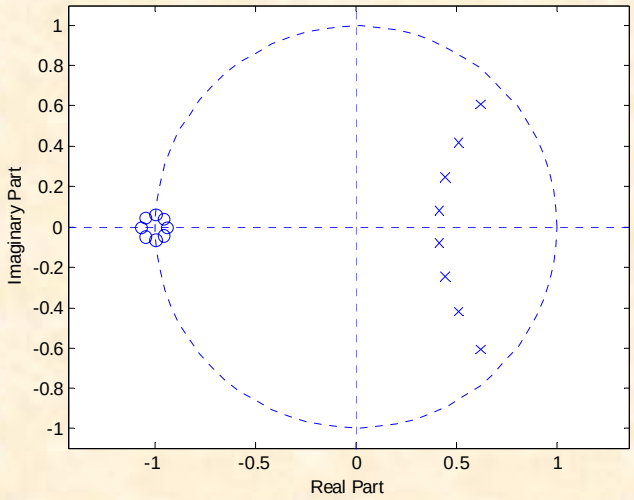


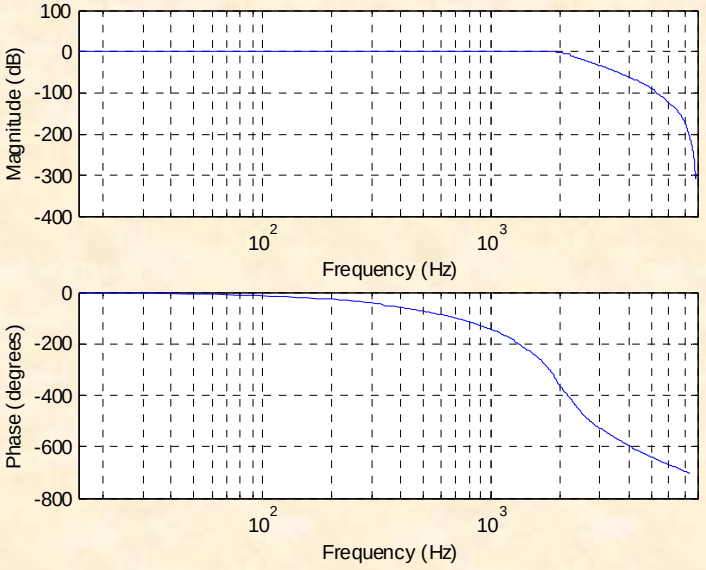


Méthode de la transformation bilinéaire









Étape 3 : structures de mise en oeuvre

- Forme directe I :

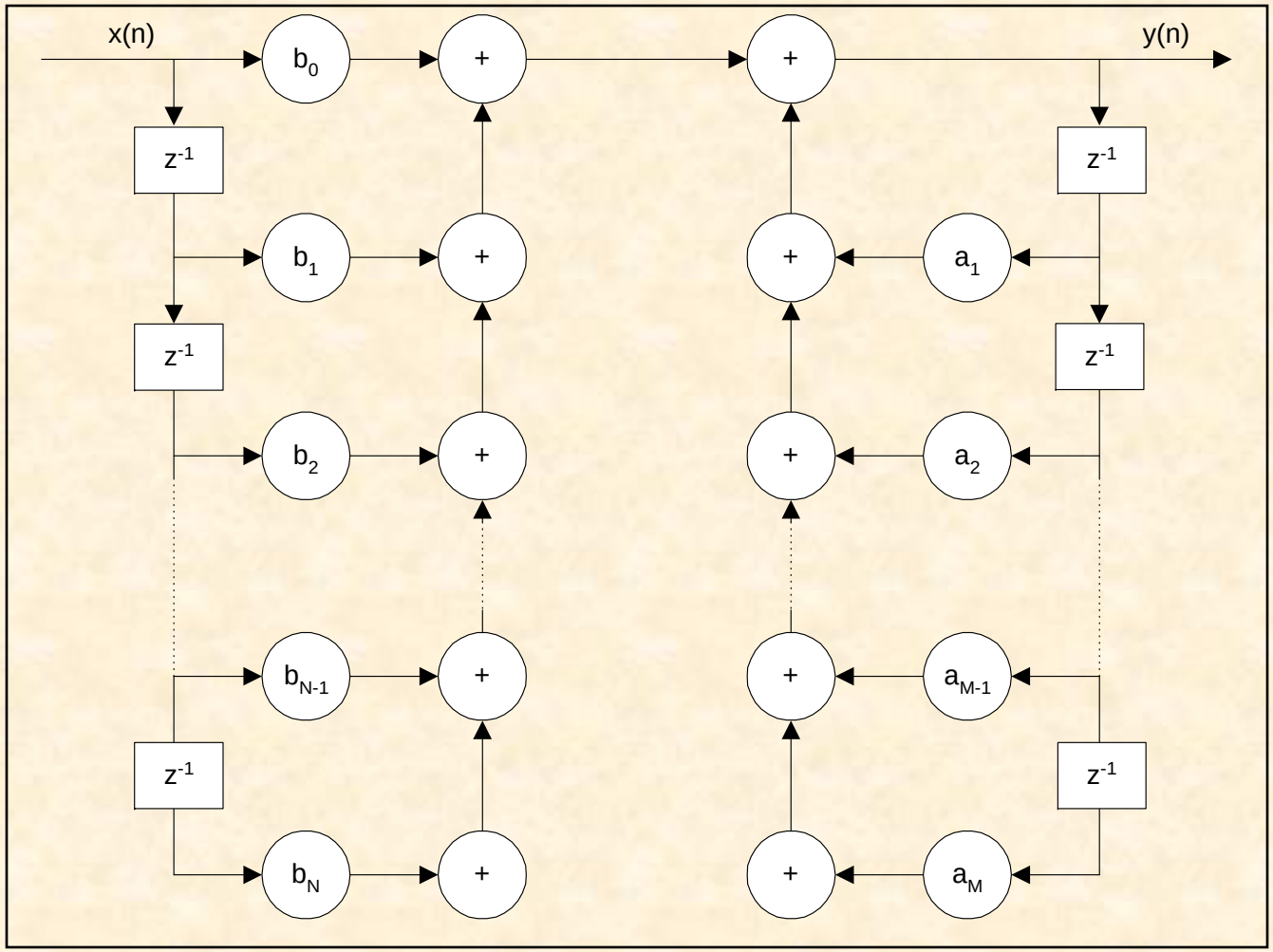
$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{k=0}^{N} b_k z^{-k}}{1+\sum_{k=1}^{M} a_k z^{-k}}=\frac{b_0+b_1 z^{-1}+\ldots+b_N z^{-N}}{1+a_1 z^{-1}+\ldots+a_M z^{-M}}$$

- Équation aux différences :

$$y[n]=\sum_{k=0}^{N} b[k]x[n-k]+\sum_{k=1}^{M} a[k]y[n-k]$$

Étape 3 : structures de mise en oeuvre

- Diagramme de flux pour la forme directe I



Étape 3 : structures de mise en oeuvre

- Forme directe II :

$$H(z) = \frac{1}{1 + \sum_{k=1}^{M} a_k z^{-k}} \sum_{k=0}^{N} b_k z^{-k} = H_1(z) H_2(z)$$

Si M=N, alors on peut poser :

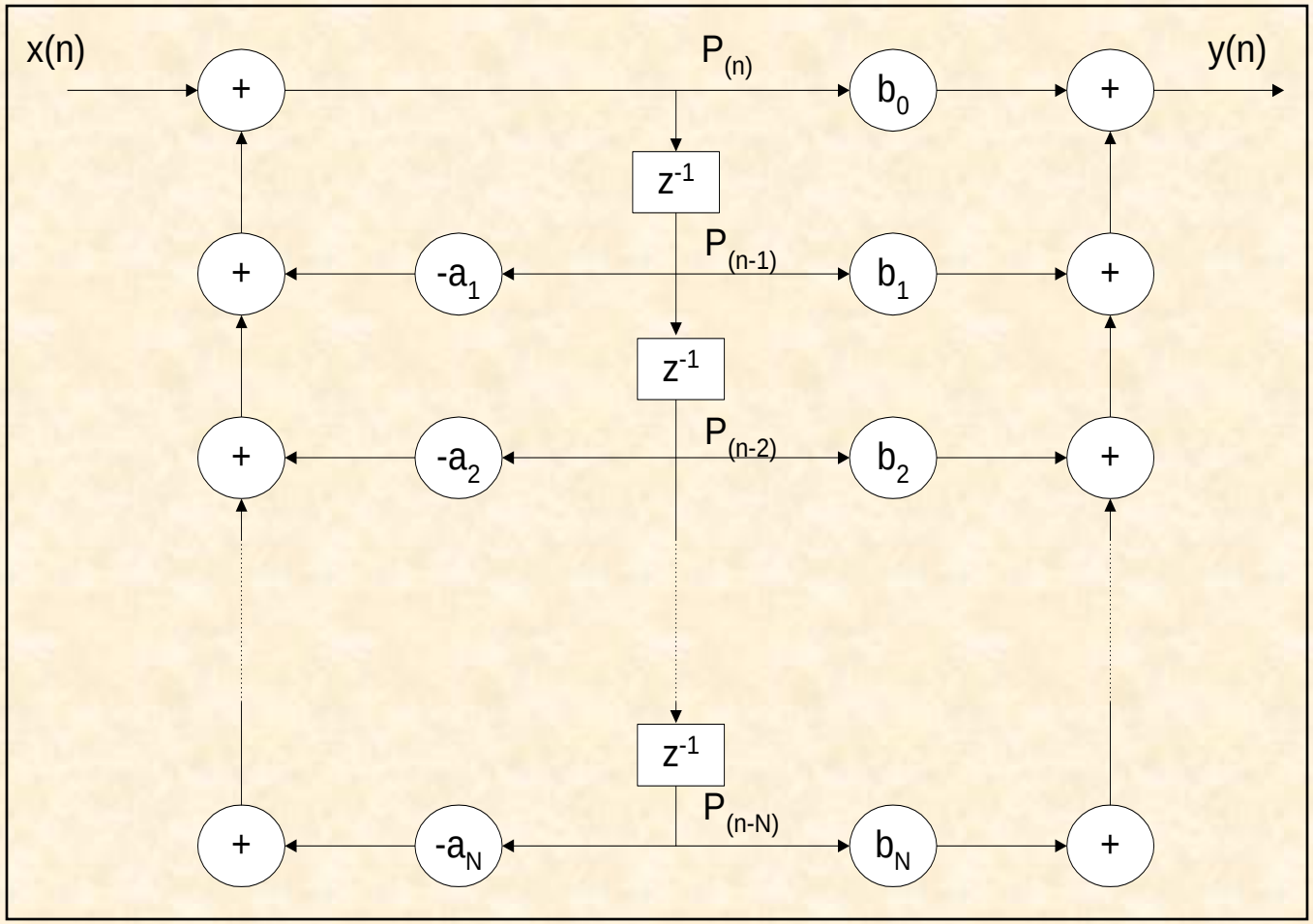
$$H_1(z) = \frac{P(z)}{X(z)} = \frac{1}{1 + \sum_{k=1}^{M} a_k z^{-k}} \quad et \quad H_2 = \frac{Y(z)}{P(z)} = \sum_{k=0}^{N} b_k z^{-k}$$

- La transformée inverse de p(z) et y(z) donne :

$$p(n) = x(n) - \sum_{k=1}^{N} a_k p(n-k) \qquad y(n) = \sum_{k=0}^{N} b_k p(n-k)$$

Étape 3 : structures de mise en oeuvre

- Diagramme de flux pour la forme directe II



Étape 5 : mise en oeuvre

Code c

```
void IIR_Isr (void)
{
      short a1 = 0x0;                              // coefficients du filtre
      short a2 = 0x15f6;
      short b0 = 0x257d;
      short b1 = 0x4afd;
      short b2 = 0x257d;

      static short p1=0, p2=0;                     // variables persistentes
      short xn, p0, y0;                            // variables d'e/s
      int prod1, prod2, prod3, prod4, prod5;       // termes intermédiaires

      xn = input_sample();
      prod1 = _mpy(p2,a2);
      prod2 = _mpy(p1,a1);
      p0 = xn + (short)((prod1 + prod2)>>15);
      prod3 = _mpy(p1,b1);
      prod4 = _mpy(p2,b2);
      prod5 = _mpy(p0,b0);
      y0 = (short)((prod3+prod4+prod5)>>15);
      p2 = p1;
      p1 = p0;

      output_sample(y0);    // Write the signal to the serial port
      return;
}
```

>>15 non requis si calculs fait en virgule flottante

Étape 5 : mise en oeuvre

```
            .def    _iir_sa
            .sect   "mycode"

_iir_sa     .cproc  a1, a2, b0, b1, b2, delays, x_ptr, y_ptr, mask, mask2

            .reg    p0, p1, p2
            .reg    prod1, prod2, prod3, prod4, prod5
            .reg    sum1, sum2, sum3
            .reg    x, y0

            LDW     *x_ptr, x
            AND     x,mask,x
            LDH     *+delays[0], p1
            LDH     *+delays[1], p2
            MPY     a1, p1, prod1
            MPY     a2, p2, prod2
            ADD     prod1, prod2, sum1
            SHR     sum1, 15, sum1
            ADD     x, sum1, p0
            MPY     b0, p0, prod3
            MPY     b1, p1, prod4
            MPY     b2, p2, prod5
            ADD     prod4, prod5, sum2
            ADD     prod3, sum2, sum3
            SHRU    sum3, 15, y0

            STH     p1, *+delays[1]
            STH     p0, *+delays[0]

            AND     y0, mask2, y0
            STW     y0, *y_ptr

            .return y0
            .endproc
```

Code assembleur linéaire