CHAPITRE 1
INTRODUCTION AUX SYSTEMES NUMERIQUES LINEAIRE ET INVARIANTS

Leonardo Pisano Fibonacci (1170-1250), mathématicien italien spécialiste de la théorie des nombres. Parmi les nombreux problèmes qu'il a traités, celui qui a donné naissance à la *séquence de Fibonacci* est resté célèbre : *« Un homme place un couple de lapins dans un enclos. Si on suppose que chaque couple donne naissance à un nouveau couple chaque mois, et que chaque nouveau couple devient lui-même productif à partir de son deuxième mois, combien de couples aura-t-il de mois en mois ? ».* Ce problème caractérise un système linéaire numérique, dont la réponse impulsionnelle est {1,1,2,3,5,8,13…}.

Ce chapitre constitue une première introduction aux systèmes numériques, et en particulier aux systèmes linéaires et invariants. Il doit permettre d'en comprendre l'extrême simplicité, sur le plan des éléments et des lois d'associations entre ces éléments, par opposition au systèmes électriques à temps continus étudiés dans un cours de théorie des circuits.

On ne fait volontairement pas mention *d'échantillons* dans ce premier chapitre (qui ne fait d'ailleurs pas mention de la variable *temps*) : les séquences numériques traitées par les filtres numériques ne sont pas exclusivement celles qui résultent de l'échantillonnage d'un signal analogique (et qui seront abordées au chapitre suivant)

L'étude précise du comportement dynamique des systèmes à temps discret est également laissée à un chapitre ultérieur, consacré à la transformée en Z.

1.1 Signaux numériques

Un signal numérique[1] *x(n)* est défini comme une séquence de nombres

[1] Ceci définit en fait un signal numérique *scalaire.* On peut également définir des signaux *vectoriels* (séquence de vecteurs) ou matriciels (séquence de matrices). Ainsi, une séquence d'images

$$\{x(n)\} = \{..., x(-n), ..., x(-1), x(0), x(1), ..., x(n), ...\} \quad (1.1)$$

Dans de nombreux cas, on manipule des séquences à partir d'une valeur de départ *x(0)* :

$$\{x(n)\} = \{x(0), x(1), ..., x(n), ...\} \quad (1.2)$$

Exemple 1.1

- *Impulsion numérique* : $\delta(n) = \begin{cases} 1 & si\ n = 0 \\ 0 & si\ n \neq 0 \end{cases}$

Ce signal est facilement obtenu dans Matlab (ici les 50 premières valeurs):

```
impulse=[1, zeros(1,49)];
```

- *Train d'impulsions numérique* : $\delta_{n_0}(n) = \begin{cases} 1 & si\ n = kn_0 \quad (k \in \mathbb{Z}) \\ 0 & si\ n \neq kn_0 \end{cases}$

Sous Matlab (ici avec n_0=10):

```
pulsetrain=[1, zeros(1,9)];
pulsetrain=[pulsetrain pulsetrain pulsetrain pulsetrain pulsetrain] ;
```

- *Echelon numérique* : $\varepsilon(n) = \begin{cases} 1 & si \quad n \geq 0 \\ 0 & si \quad n < 0 \end{cases}$

```
step=[ones(1,50)];
```

- *Exponentielle imaginaire numérique* : $f(n) = e^{jn\varphi}$

Sous Matlab (ici avec φ=pi/20):

```
imagexp=[exp(j*(0:49)*pi/20)];
```

La Fig. 1.1 montre comment Matlab permet d'afficher ces séquences numériques de base :

```
Subplot(2,2,1); stem(impulse);
Subplot(2,2,2); stem(pulsetrain);
Subplot(2,2,3); stem(step);
Subplot(2,2,4); stem(real(imagexp)); hold on; stem(imag(imagexp));
```

---

numériques est un signal numérique matriciel. On parle également de signaux *mono-* et *multi-dimensionnels.* Une image unique peut être vue comme un signal bi-dimensionnel.

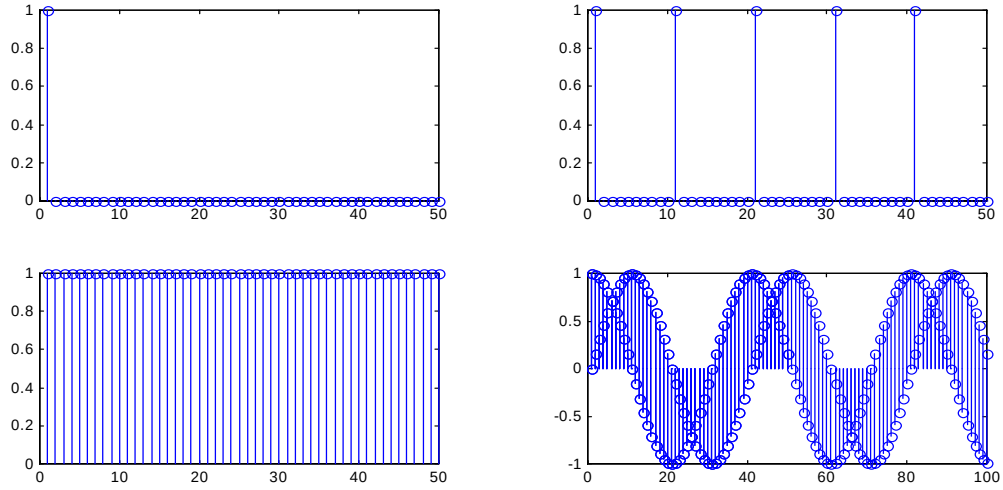


Fig. 1.1 Séquences numériques de base

1.2 Systèmes numériques

Un système numérique (Fig. 1.2) reçoit en entrée une séquence de nombres $\{x(0), x(1), x(2), \ldots\}$, notée plus simplement $x(n)$, et produit en sortie une séquence de nombres $y(n)$ obtenue à partir de l'entrée après application d'un algorithme[2].

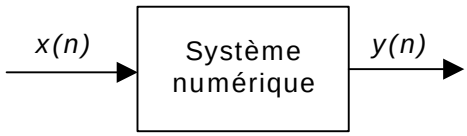


Fig. 1.2 Système numérique

Un système numérique est *linéaire et invariant* s'il répond en outre aux conditions classiques illustrées à la Fig. 1.3.

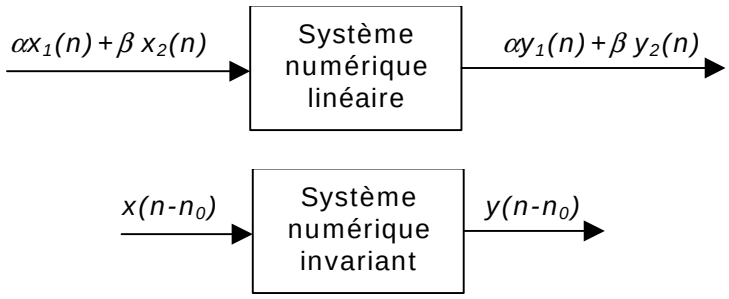


Fig. 1.3 Système numérique linéaire et invariant

[2] En pratique un système peut avoir plusieurs entrées et/ou plusieurs sorties. On parle de système SISO (Single Input – Single Output) ou MIMO (Multiple Inputs – Multiple Outputs).

Un système numérique est *stable* si, lorsqu'on lui présente une entrée finie, il produit une sortie finie.

Exemple 1.2

Le système défini par la relation $y(n)=Kx(n)+A$ (où K et A sont des constantes réelles) est non linéaire, invariant, et stable.

L'élément « délai » défini par la relation $y(n)=x(n-1)$ est linéaire, invariant, et stable.

Le système défini par la relation $y(n)=n\,x(n)$ est linéaire, non invariant, et instable.

Le système défini par la relation $y(n)=|x(n)|^2$ est non linéaire, invariant, et stable.

1.3 Systèmes récursifs – Systèmes non récursifs

D'une manière générale, les systèmes numériques linéaires et invariants (SLI) peuvent être décrits par des équations aux différences finies, linéaires et à coefficients constants, de la forme :

$$\begin{aligned} y(n)+a_1y(n-1)+a_2y(n-2)+\ldots+a_Ny(n-N) \\ =b_0x(n)+b_1x(n-1)+\ldots+b_Mx(n-M) \end{aligned} \qquad (1.3)$$

Une telle équation exprime le fait que la valeur de la sortie $y(n)$ à l'instant courant est une combinaison linéaire des N sorties précédentes, de l'entrée courante, et des M entrées précédentes. On la note souvent de façon plus compacte :

$$y(n)+\sum_{i=1}^{N}a_iy(n-i)=\sum_{i=0}^{M}b_ix(n-i) \qquad (1.4)$$

De tels systèmes sont dits *récursifs* étant donnée le caractère récursif de l'équation correspondante : il faut avoir déjà calculé toutes les sorties précédentes pour pouvoir calculer la sortie courante.

Un cas particulier est celui des systèmes dont les valeurs de sortie ne dépendent que de l'entrée et de son passé :

$$y(n)=\sum_{i=0}^{M}b_ix(n-i) \qquad (1.5)$$

De tels systèmes sont dits *non récursifs.*

L'*ordre* d'un SLI numérique est donné par le degré de la récursivité de l'équation aux différences finies associée : *N.*

De la même façon qu'une équation intégro-différentielle entre l'entrée et la sortie (toutes deux analogiques) d'un système à temps continu définit un filtre analogique, on appelle *filtre numérique* un système numérique linéaire et invariant. On parlera donc dans la suite de *filtre récursif* et de *filtre non récursif*.

Il est possible de visualiser l'équation de récurrence associée à un filtre numérique, sous la forme d'une *structure* (ou *graphe de fluence)* faisant apparaître les éléments de base suivants :

- L'additionneur, symbolisé par  , qui additionne les signaux à ses entrées.
- Le multiplieur, symbolisé par  , qui multiplie un signal par un scalaire *a*

- L'élément « délai », symbolisé par $\boxed{z^{-1}}$, qui produit une sortie retardée de une valeur par rapport à son entrée.

La structure associée à (1.3) est donnée à la Fig. 1.4 [3]

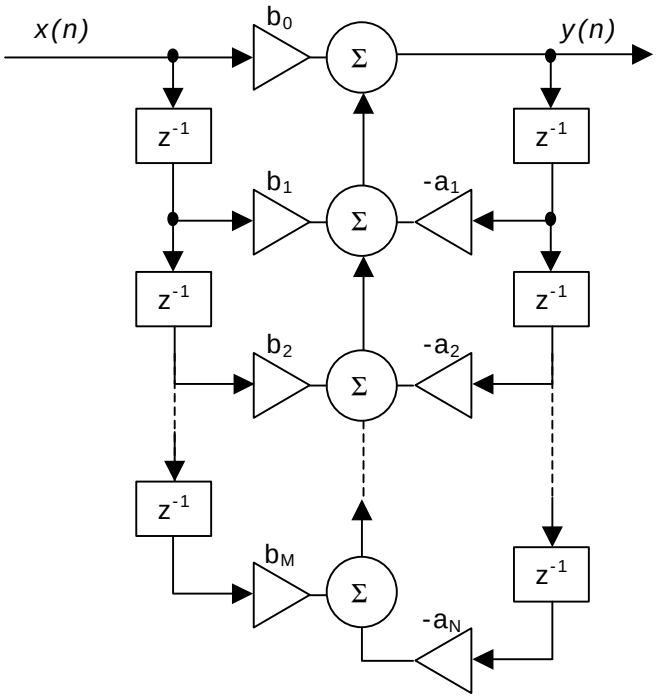


Fig. 1.4 Exemple de structure d'un filtre numérique récursif

On constate que le caractère récursif ou non récursif d'un filtre numérique est lié à la présence ou à l'absence de boucles dans sa structure.

Exemple 1.3

Le principe des intérêts composés sur un compte bancaire peut être facilement défini par une équation aux différences finies :

$$y(n) = y(n-1) + \frac{p}{100} y(n-1) + x(n) \qquad (1.6)$$

où $y(n)$ est la somme disponible au temps $t=nT$ sur le compte, $x(n)$ est le dépôt ou le retrait courant et p est le taux d'intérêt. La valeur de T dépend de la politique financière de l'établissement (actualisation par jour, semaine, mois, année…).

Le système sous-jacent est récursif et d'ordre 1. Sa structure est donnée à la Fig. 1.5.

[3] Plus précisément, il s'agit là d'une des structures possibles. Nous verrons au chapitre consacré au filtrage numérique que plusieurs structures existent, avec chacune leurs avantages et leurs inconvénients.

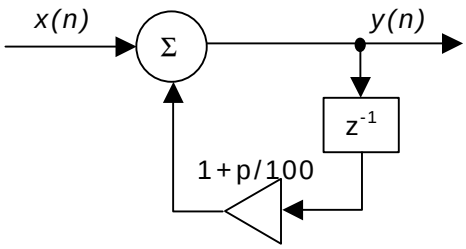


Fig. 1.5 Structure du SLI correspondant au calcul des intérêts composés

Exemple 1.4

La séquence de Fibonacci peut être définie par une équation aux différences finies :

$$y(n) = y(n-1) + y(n-2) + x(n) \qquad (1.7)$$

Où y(n) est le nombre de couples de lapins dans l'enclos au $n^{ème}$ mois et *x(n)* est le nombre de couples que l'on ajoute à la population de l'enclos au $n^{ème}$ mois.

Le système sous-jacent est récursif et d'ordre 2. Sa structure est donnée à la Fig. 1.6

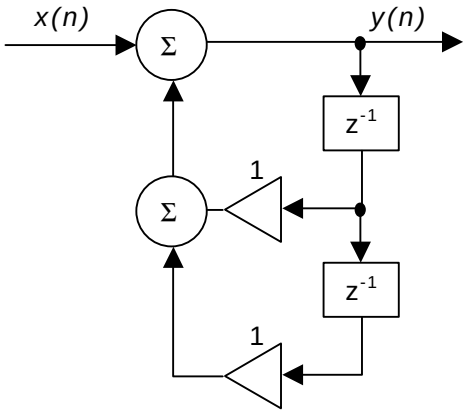


Fig. 1.6 Structure du SLI correspondant au calcul de la séquence de Fibonacci

Exemple 1.5

Imaginons que l'on veuille créer un filtre numérique dont chaque valeur de sortie soit la moyenne locale du signal d'entrée, estimée sur $2L+1$ valeurs autour de l'échantillon courant :

$$y(n) = \frac{1}{2L+1} \sum_{i=-L}^{L} x(n-i) \qquad (1.8)$$

L'équation précédente définit bien un filtre numérique non récursif, appelé *filtre à moyenne mobile*.

Notons que ce filtre, tel que défini, est *non causal* : le calcul de la valeur courante de sortie courant nécessite de connaître les *L* valeurs d'entrée *suivantes.* L'existence de systèmes non causaux n'est pas nécessairement un problème dans le domaine du numérique : ces systèmes seront en pratique implémentés comme des systèmes causaux *à délai,* le délai correspondant au nombre de valeurs futures nécessaires au calcul de la valeur de sortie courante. Le système causal à délai correspondant à l'équation précédente est donné par la Fig. 1.7.

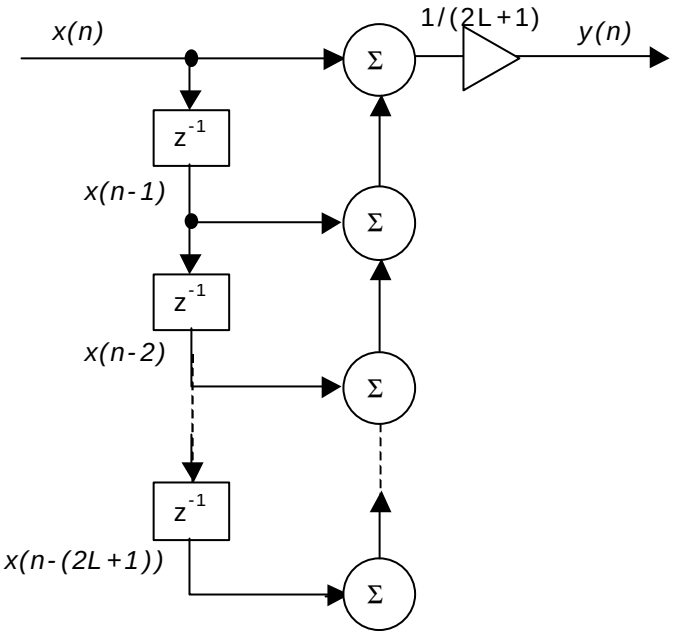


Fig. 1.7 Structure du SLI causal à délai (*L*) correspondant au filtre à moyenne mobile

1.4 Réponse impulsionnelle

La *réponse impulsionnelle,* notée *h(n)*, d'un SLI numérique est alors définie comme sa réponse forcée *y(n)* à l'impulsion numérique (Fig. 1.8), en supposant que les conditions initiales sont nulles : $y(n)=0$ pour $n<0$.

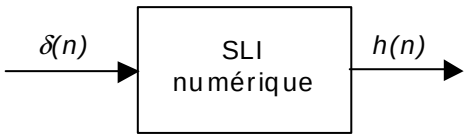


Fig. 1.8 Réponse impulsionnelle

Au contraire des systèmes analogiques, le calcul de la réponse impulsionnelle d'un SLI numérique est immédiat : il suffit d'effectuer la récurrence numérique sous-jacente[4].

Exemple 1.6

La réponse impulsionnelle du système des intérêts composés décrit par (1.6) est :

$h(n)=\{1, 1+p/100, (1+p/100)^2, (1+p/100)^3, \ldots\}$

On peut l'obtenir sous Matlab grâce à la fonction `filter`, qui implémente la récurrence générale (1.4). Le résultat est affiché à la Fig. 1.9 (on suppose ici que *p* vaut 5):

```
impresp=filter([1],[1 -1.05],impulse); stem(impresp);
```

[4] Ceci illustre bien ce qui avait été annoncé dans l'introduction de ce cours : *simuler un traitement numérique, c'est l'effectuer*.

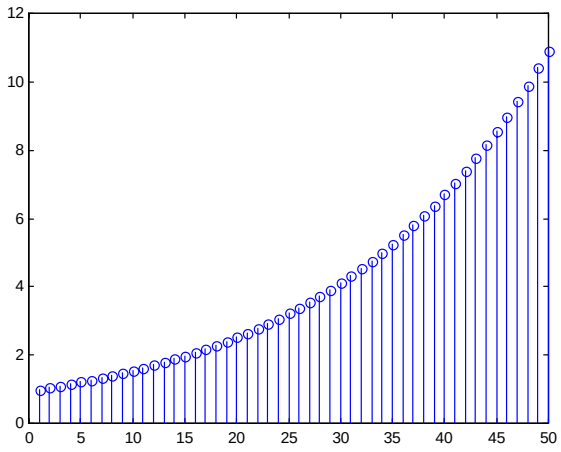


Fig. 1.9 Réponse impulsionnelle du filtre correspondant au calcul des intérêts composés

On constate en passant que ce système est instable (mais qui s'en plaindra ?), puisque sa réponse à une séquence de nombre finis produit une séquence de nombres qui tendent vers l'infini.

Exemple 1.7

La séquence de Fibonacci est précisément la réponse impulsionnelle du système répondant à (1.7), puisqu'on suppose commencer l'élevage avec un couple de lapins et ne plus en ajouter par la suite : $h(n) = \{1, 1, 2, 3, 5, 8, 13, 21, \ldots\}$

Sous Matlab :

```
impresp=filter([1],[1 –1 -1],impulse);
impresp(1:12)
    1     1     2     3     5     8    13    21    34    55    89   144
stem(impresp);
```

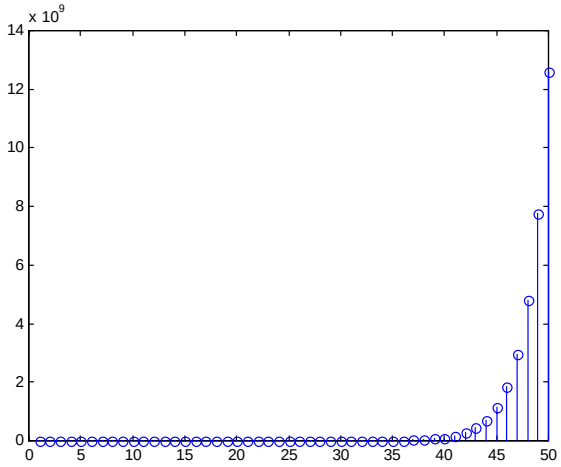


Fig. 1.10 Réponse impulsionnelle du filtre correspondant à la séquence de Fibonacci

---

[5] La fonction `stem` permet de visualiser une séquence numérique sans interpolation linéaire entre les échantillons successifs, ce qui permet de ne pas la confondre avec un signal analogique.

Comme on pouvait s'y attendre, ce système est également instable ☺

Exemple 1.8

L'examen de la structure du filtre à moyenne mobile suffit à se convaincre que sa réponse impulsionnelle est donnée par la séquence {1, 1, 1, …, 1, 0, 0, 0, …} comprenant *2L+1* « 1 » et une infinité de « 0 ».

On notera au passage que *la réponse impulsionnelle d'un filtre non récursif n'est rien d'autre que la suite de ses coefficients* $\{b_0, b_1, b_2, ,\ldots\}$, qui se trouvent successivement poussés vers la sortie par l'impulsion qui se propage le long de la chaîne d'éléments à délai.

Sous Matlab (avec $L = 3$):

```
impresp=filter(1/7*[1 1 1 1 1 1 1],[1],impulse); stem(impresp);
```

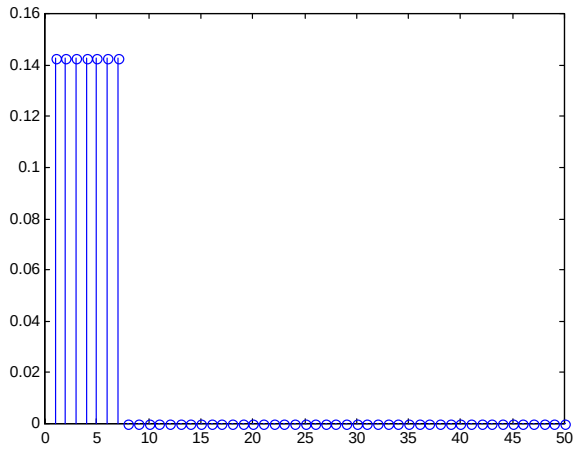


Fig. 1.11 Réponse impulsionnelle du filtre à moyenne mobile causal à délai

Nous pouvons donc penser que ce filtre est stable, bien que rien ne nous permette pour le moment d'affirmer que la réponse impulsionnelle soit suffisante pour tester la stabilité d'un filtre. Nous verrons plus tard que c'est bien le cas.

On constate sur les exemples précédents que la réponse impusionnelle d'un filtre récursif est une séquence illimitée de valeurs non identiquement nulles (même si le filtre est stable), puisque chaque valeur de sortie dépend des valeurs de sorties précédentes. Par contre, la réponse impulsionnelle d'un filtre non récursif est donnée par la suite de ses coefficients, et constitue donc une séquence qui s'annule après un nombre fini de valeurs. Il s'ensuit que les filtres récursifs sont appelés *filtres à réponse impulsionnelle infinie (RII)* et que les filtres non récursifs sont dits *à réponse impulsionnelle finie (RIF)*.

1.5 Réponse forcée à une entrée quelconque – Convolution numérique

Le calcul de la réponse à une entrée quelconque est tout aussi trivial que celui de la réponse impulsionnelle : il suffit d'effectuer la récurrence numérique.

Il est cependant intéressant de constater que la réponse à une entrée quelconque peut être déterminée facilement si on connaît déjà la réponse impulsionnelle. On peut en effet toujours considérer qu'une séquence d'entrée

$\{x(n)\}=\{x(0),\ x(1),\ x(2),\ \ldots\}$ est une somme d'impulsions numériques pondérées et décalées :

$$x(n)=x(0)\delta(n)+x(1)\delta(n-1)+x(2)\delta(n-2)+\ldots \quad \text{(pour tout } n\text{)} \qquad (1.9)$$

Vu les propriétés de linéarité et d'invariance du système, la réponse à $x(n)$ est donnée par :

$$y(n)=x(0)h(n)+x(1)h(n-1)+x(2)h(n-2)+\ldots \qquad (1.10)$$

ce qui n'est rien d'autre qu'une combinaison linéaire des réponses impulsionnelles à chacun des nombres de la séquence d'entrée (et donc décalées d'autant).

Cette équation définit ce que l'on appelle une *convolution numérique,* notée * :

$$y(n)=x(n)*h(n)=\sum_{i=-\infty}^{\infty}x(i)h(n-i) \qquad (1.11)$$

On notera que, dans cette dernière équation, la sommation est définie pour i allant de $-\infty$ à $+\infty$, et non de 0 à $+\infty$ comme on pourrait s'y attendre vu (1.10). Cette extension permet de prendre en compte les systèmes non causaux, dont la réponse impulsionnelle $h(n)$ peut être définie pour n négatif.

Exemple 1.9

Considérons à nouveau le système de Fibonacci et supposons que le propriétaire de l'enclos y apporte, non pas un seul couple de lapins le premier mois, mais bien : un couple le premier mois, deux couples le troisième, et un couple le quatrième mois. On demande combien de couples se trouveront dans l'enclos après sept mois.

Il est facile de lancer le calcul de la récurrence sous Matlab :

```
resp=filter([1],[1 –1 -1],[1,0,2,1,0,0,0])
     1     1     4     6    10    16    26
```

On peut retrouver facilement ce résultat en utilisant la réponse impulsionnelle {1,1,2,3,5,8,13,21,34,55,89,144} : le premier couple de lapins produira 13 couples en sept mois, les deux couples placés au troisième mois en produiront 2*5, et le quatrième en produira 3, ce qui fait bien en tout 26.

On peut montrer l'équivalence des deux approches sous Matlab :

```
impresp=filter([1],[1 –1 -1],impulse(1 :7)); % calcul par la réponse impulsionnelle
subplot(5,1,1) ; stem(impresp); v=axis;
subplot(5,1,2) ; stem([0 0 2*impresp(1:5)]); axis(v);
subplot(5,1,3) ; stem([0 0 0 impresp(1:4)]); axis(v);
subplot(5,1,4) ; stem(impresp+[0 0 2*impresp(1:5)]+[0 0 0 impresp(1:4)]);
subplot(5,1,5) ; stem(resp); % calcul par recurrence, pour comparaison
```

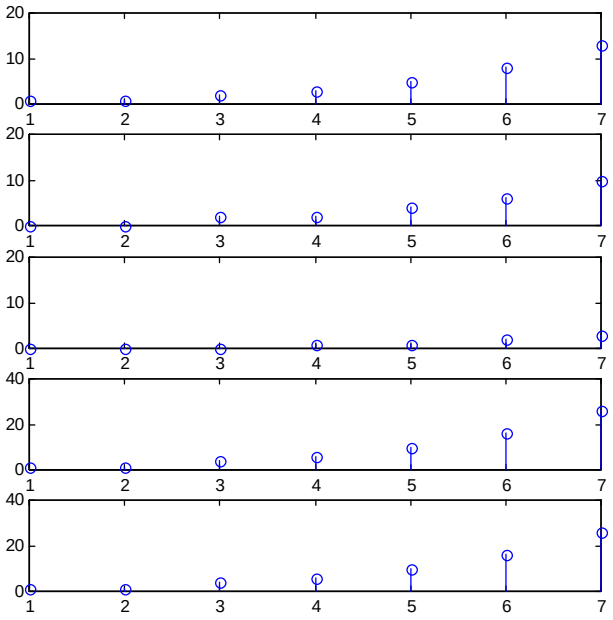


Fig. 1.12 Somme de réponses impulsionnelles décalées, comparées à la réponse d'un filtre

On montrera à titre d'exercice que le produit de convolution ainsi défini est bien commutatif :

$$y(n) = x(n) * h(n) = \sum_{i=0}^{\infty} h(i)x(n-i) \qquad (1.12)$$

Cette façon d'écrire le produit de convolution possède elle aussi une interprétation simple : pour obtenir une valeur particulière $y(n_0)$ de la séquence $y(n)$, il suffit d'inverser $h(n)$, de positionner $h(0)$ sur $x(n_0)$, et de calculer le produit scalaire entre les séquences $x(n)$ et $h(n)$ ainsi définies. On constate ainsi qu'il est possible d'obtenir directement une valeur $y(n_0)$ de la séquence $y(n)$ par combinaison linéaire des valeurs de $x(n)$ autour de $x(n_0)$, combinaison linéaire dont les coefficients sont les valeurs de $h(n)$ (Fig. 1.13).

Exemple 1.10

Soit $\{x(n)\} = n$ et $\{h(n)\} = \{1,1/2,1/3,1/4,0,0,\ldots\}$. Le calcul de $y(6)$ par (1.12) donne 10.5833. Cette valeur correspond à 6+5/2+4/3+3/4 (Fig. 1.13).

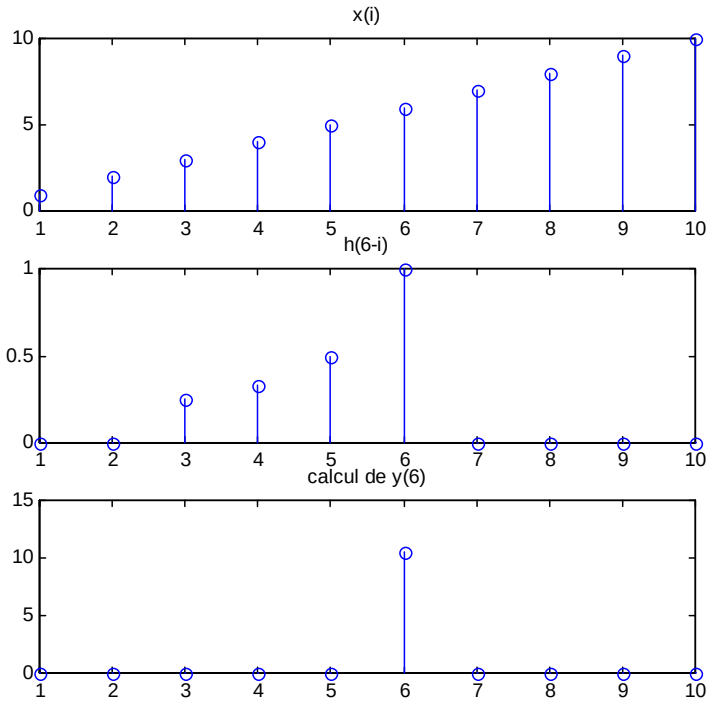


Fig. 1.13 Calcul de $y(6)$ par combinaison linéaire des valeurs de x autour de $x(6)$

Pour rappel, l'équation de récurrence (1.4) nécessite le calcul de toutes les valeurs de $y(n)$ précédant $y(n_0)$[6]. Ce calcul est réalisé ici indirectement par le calcul de $h(n)$. En pratique, nous verrons que la réponse impulsionnelle des filtres à réponse impulsionnelle infinie peut prendre des valeurs importantes pour n très grand. Le calcul par (1.4) est donc souvent préférable à (1.12). A fortiori, le calcul par (1.12) est impossible pour les systèmes instables.

Le calcul de la réponse à une entrée quelconque est tout aussi trivial que celui de la réponse impulsionnelle : il suffit d'effectuer la récurrence numérique.

1.6 Transformée en *Z* – Pôles et zéros d'un SLI

Les opérations effectuées pour réaliser la convolution entre deux séquences ne sont pas sans rappeler celles qui permettent de calculer un produit de polynômes. Le produit de deux polynômes se réduit en effet à une somme de produits des coefficients du premier par les coefficients du second, décalés des puissances correspondantes. On montre facilement que ces deux opérations sont exactement identiques : la convolution entre deux séquences numériques est exactement équivalente au produit de deux polynômes dont les coefficients sont précisément les séquences à convoluer.

Exemple 1.11

---

[6] Sauf si le système est non-récursif, auquel cas on a déjà vu que sa réponse impulsionnelle est égale à la suite de ses coefficients. Le calcul de $y(n_0)$ par (1.5) est alors exactement identique à (1.12).

Soit $\{x(n)\}=\{1,2,3,0,0,\ldots\}$ et $\{h(n)\}=\{2,3,4,0,0,\ldots\}$. La convolution de $\{x(n)\}$ avec $\{h(n)\}$ par (1.11) donne $\{2,7,16,17,12,0,0,\ldots\}$. Construisons deux polynômes $X(z)$ et $H(z)$ à partir de $\{x(n)\}$ et $\{h(n)\}$:

$X(z)=1+2z^{-1}+3z^{-2}=z^{-2}(z^2+2z+3)$ $\quad$ $H(z)=2+3z^{-1}+4z^{-2}=z^{-2}(2z^2+3z+4)$

Le produit de ces deux polynômes donne bien le polynôme $Y(z)=z^{-4}(2z^4+7z^3+16z^2+17z+12)$

On peut également le vérifier par Matlab :

```
conv([2,3,4],[1 2 3])
ans =
     2     7    16    17    12
```

On comprend mieux maintenant pourquoi, dans Matlab, la fonction qui réalise le produit de deux polynômes s'appelle conv.

L'exemple précédent a montré qu'il est intéressant d'associer à une séquence numérique $\{x(n)\}$ un polynôme en z^{-1} (où z est la variable complexe), que l'on appelle *transformée en Z* de $\{x(n)\}$ et que l'on note $X(z)$:

$$\{x(n)\} \overset{Z}{\Leftrightarrow} X(z)=\sum_{i=-\infty}^{\infty} x(i)z^{-i} \tag{1.13}$$

Si $H(z)$ est la transformée en z de la réponse impulsionnelle $\{h(n)\}$ d'un SLI numérique :

$$\{h(n)\} \overset{Z}{\Leftrightarrow} H(z)=\sum_{i=-\infty}^{\infty} h(i)z^{-i} \tag{1.14}$$

La transformée en z de la sortie du système est alors donnée par :

$$\{y(n)\} \overset{Z}{\Leftrightarrow} Y(z)=X(z)H(z) \tag{1.15}$$

On voit donc que la transformée en z (qui associe une fonction de la variable complexe z à un signal numérique) est l'homologue, dans le domaine numérique, de la transformée de Laplace (qui associe une fonction de la variable complexe p à un signal analogique) :

$$x(t) \overset{L}{\Leftrightarrow} X(p)=\int_{-\infty}^{\infty} x(t)e^{-pt}dt \tag{1.16}$$

Le passage de l'analogique au numérique correspond au passage de l'intégrale à la somme, et la substitution de z à e^p.

Comme la transformée de Laplace, la transformée en z ne converge pas nécessairement pour toutes les valeurs de z. Elle jouit également de nombreuses propriétés, dont on trouvera un exposé à l'annexe I. De même, on peut dresser une liste des valeurs de $X(z)$ pour certains signaux simples. L'équation (1.15) permet alors, si l'on connaît $X(z)$ et $H(z)$, de calculer $Y(z)$, et d'en déduire $\{y(n)\}$. On peut d'ailleurs également utiliser pour ce faire la décomposition en fractions simples.

La transformée en z est cependant largement moins indispensable en numérique que son homologue de Laplace ne l'est en analogique. La récurrence (1.4), qui se substitue à l'équation intégro-différentielle décrivant un SLI analogique, permet en effet le calcul direct de la sortie, sans devoir passer par le calcul de sa transformée en z.

La seule propriété fondamentale de la transformée en *z* que nous utiliserons ici est celle liée au retard de $\{x(n)\}$ de n_0 échantillons:

$$\{x(n-n_0)\} \overset{Z}{\Leftrightarrow} \sum_{i=-\infty}^{\infty} x(i-n_0)z^{-i} = \sum_{k=-\infty}^{\infty} x(k)z^{-(k+n_0)} = z^{-n_0}\sum_{k=-\infty}^{\infty} x(k)z^{-k} = z^{-n_0}X(z) \quad (1.17)$$

On peut en effet avantageusement appliquer cette propriété à la récurrence (1.4) :

$$Y(z)+Y(z)\sum_{i=1}^{N} a_i z^{-i} = X(z)\sum_{i=0}^{M} b_i z^{-i} \quad (1.18)$$

ce qui conduit à une autre formulation de *H(z)* :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1+\sum_{i=1}^{N} a_i z^{-i}} \quad (1.19)$$

que l'on réécrit parfois sous la forme plus simple :

$$H(z) = \frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{i=0}^{N} a_i z^{-i}} \quad (a_0 = 1) \quad (1.20)$$

Cette dernière équation définit la fonction de transfert en *z* d'un SLI, et est à comparer à l'équation donnant la forme de la fonction de transfert en *p* d'un système analogique. On définit les *zéros* et les *pôles* d'un SLI numérique comme les racines du numérateur et du dénominateur de *H(z).*

Enfin, on peut établir un lien entre la position des pôles d'un système numérique et sa stabilité, et montrer[7] qu'un SLI numérique est strictement stable si ses pôles sont tous à l'intérieur du cercle de rayon unité (cercle non compris), et stable si on accepte aussi les pôles simples[8] sur le cercle de rayon unité. Le demi-plan de droite se trouve ainsi transformé en l'intérieur du cercle de rayon unité, ce qui correspond à la substitution de *z* à e^p déjà évoquée plus haut (Fig. 1.14).

---

[7] La démonstration en est assez lourde.

[8] Rappelons qu'en analogique un pôle multiple correspond à un terme $t\ exp(-\sigma t)\ cos(\omega t)$. Un pôle multiple sur l'axe imaginaire conduit donc à une réponse non bornée. Pour les mêmes raisons, un pôle multiple sur le cercle de rayon unité correspond à un système numérique instable.

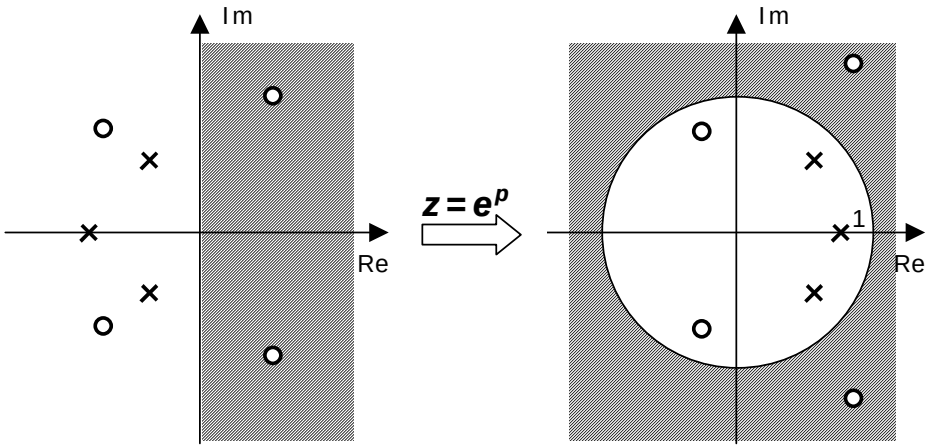


Fig. 1.14 Zone de stabilité en *z* et lien avec la zone de stabilité en *p*.

Exemple 1.12

Nous avions déjà du que le système de Fibonacci est instable. Nous pouvons maintenant le vérifier sur sa fonction de transfert :

$$H(z) = \frac{1}{1 - z^{-1} - z^{-2}}$$

Les pôles de *H(z)* sont réels et égaux à $(1 \pm \sqrt{5})/2$. On peut les afficher facilement sous Matlab, qui les positionne automatiquement par rapport au cercle de rayon unité :

```
zplane([1],[1 -1 -1])
```

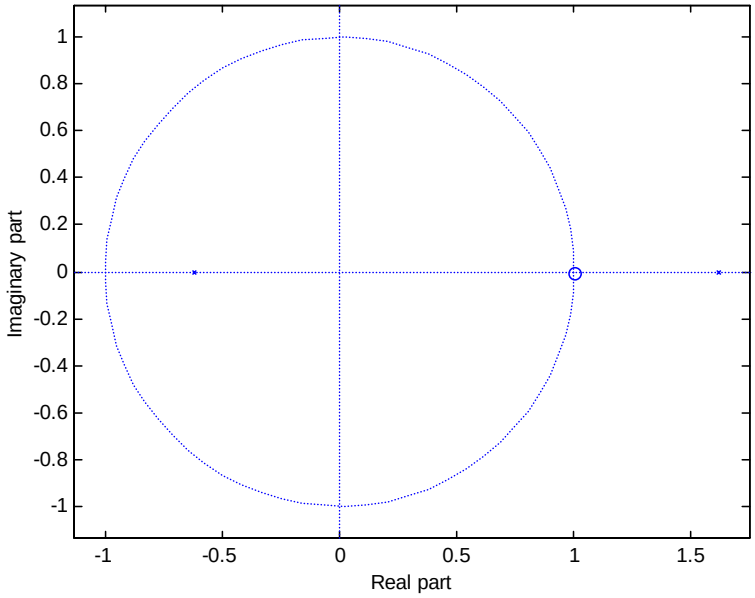


Fig. 1.15 Affichage des pôles et zéros par Matlab (le zéro est une erreur de Matlab)

Exercices

Exercice 1.1

Un SLI est décrit par l'équation :

$$y(n)+a_1y(n-1)=b_0x(n)+b_2x(n-2)$$

On demande :

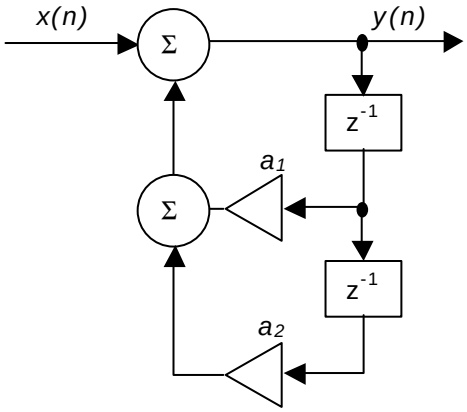
- De dessiner la structure du filtre correspondant
- De trouver l'expression *analytique* de sa réponse impulsionnelle
- D'en déduire une condition pour que le filtre soit stable
- De vérifier cette condition à partir de la fonction de transfert en *z* du système.

Solution

$h(n)=h_1(n)+h_2(n)=b_0(-a_1)^n\varepsilon(n)+b_2(-a_1)^{n-2}\varepsilon(n-2)$, stable si $|a_1|<=1$

Exercice 1.2

Le système suivant constitue, pour certaines valeurs de ses coefficients, un *oscillateur numérique.* Il implémente un calcul purement récursif de la fonction cos(x).



On demande :

- D'écrire la récurrence numérique correspondante
- De trouver les les valeurs de a_1 et a_2 permettant à ce filtre d'avoir une réponse impulsionnelle possédant une composante de régime[9] en $\{\cos(n\varphi)\}$
- De calculer et d'afficher la réponse impulsionnelle sous Matlab
- De calculer les pôles de ce système et de les dessiner dans le plan complexe.

Solution

$a_1=-2\cos(\varphi);a_2=1$; pôles complexes = ($e^{j\varphi}$, $e^{-j\varphi}$)

Exercice 1.3

En utilisant conv, calculer et afficher sous Matlab les réponses indicielles des filtres donnés par les réponses impulsionnelles suivantes :

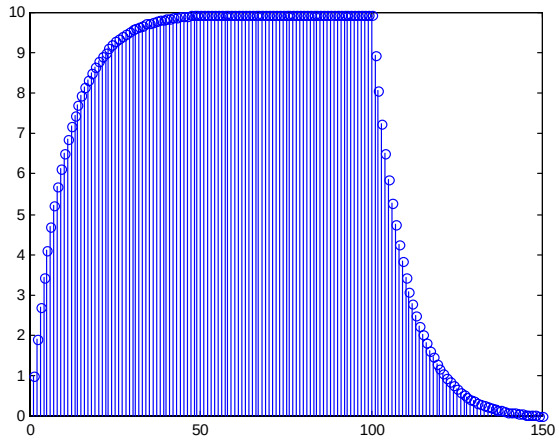
- Passe-bas : $h(n)=0.9^n\,\varepsilon(n)$
- Passe-haut : $h(n)=(-0.9)^n\,\varepsilon(n)$
- Passe-bande : $h(n)=2(0.9)^n\cos(n\pi/2)\,\varepsilon(n)$

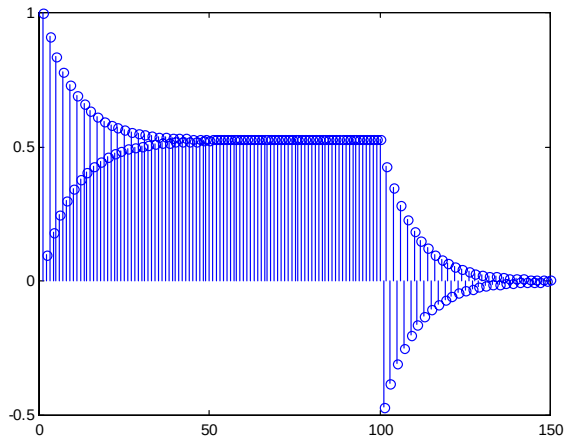
Ces convolutions auraient pu être exécutées par la fonction `filter`. Comment ?

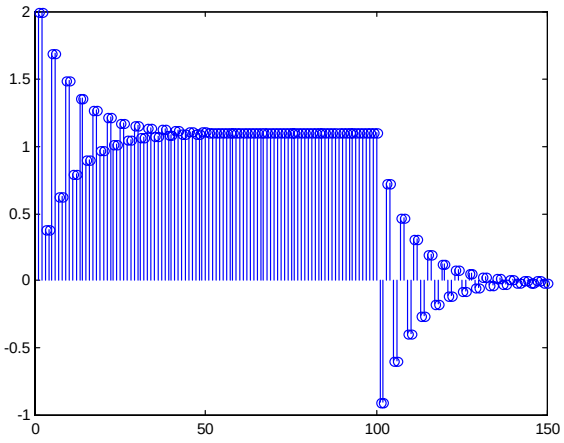
Solutions[10]

---

[9] On supposera donc que *y(n)* est défini même pour $n<0$.








```
filter(h,1,ones(1,100))) ;
```

---

[10] Les affichages qui suivent ont été obtenus sur base des 50 premières valeurs de l'entrée indicielle. La deuxième partie des graphiques correspond aux valeurs suivantes, pour lesquelles Matlab a supposé implicitement que l'excitation était nulle. Ceci fait apparaître une deuxième réponse indicielle, inversée.