UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des Sciences de l'Ingénieur

Département d'Electronique

MEMOIRE DE MAGISTER

Spécialité : Signaux et systèmes

SYNTHESE DE FILTRES NUMERIQUES EN PRECISION FINIE PAR LES ALGORITHMES GENETIQUES

Par

Miloud DJAOUANE

Devant le jury composé de :

H. SALHI	Maître de conférences, U. de Blida	Président
M. ALLEK	Maître de recherche, CRND/COMENA	Examinateur
Z. BENSLAMA	Maître de conférences, U. de Blida	Examinateur
A. GUESSOUM	Professeur, U. de Blida	Rapporteur
A. BENOUARED	Chargé de cours, U. de Blida	Co-Rapporteur

Blida, Avril / 2010

RESUME

L'implantation d'applications de traitement numérique du signal dans un système embarqué requiert l'usage de l'arithmétique à virgule fixe et de minimiser le nombre de bits pour représenter les données. Cette action nécessite de travailler en précision finie afin de réduire la surface et la consommation, sans entraîner une perte de précision des calculs; De ce fait, on proposera dans ce travail, une méthode d'optimisation à base d'algorithmes génétiques fonctionnant en précision finie, dans un espace de recherche quantifié, appliquée à la synthèse de filtres numériques à coefficients réels codés en virgule fixe. Les travaux de simulations réalisés, montrent l'efficacité et l'importance de cette démarche en termes d'optimisation et de synthèse de filtres numériques en temps différé.

ملخص

تحتاج تطبيقات المعالجة الرقمية للإشارة داخل الأنظمة المدمجة إلى المعالجة العددية التي تعتمد على الفاصلة الثابتة و أيضا إلى التقليل من عدد البيتات الممثلة للمعطيات, هذا العمل يحتاج إلى الحسابات ذات الدقة المنتهية و هذا للتقليل من حجم مساحة الدارات المستعملة, دون المساس بدقة الحسابات.

نقترح من خلال هذا المشروع طريقة ذات دقة متناهية، تعتمد على الخوارزميات الجينية و ذالك داخل وسط كمي مطبق على تحليل المرشحات الرقمية ذات المعاملات الحقيقية و الترميز بالفاصلة الثابتة. تثبت محاكاة المشروع التي انجزناها نجاعة و أهمية الطريقة المقترحة.

ABSTRACT

The constraints compulsory for the real-time applications of the embarked systems are on one hand the reduction of surface and on the other hand the reduction of the consumption. The conception in finished precision allows optimizing these constraints. Therefore, we will propose in this work, a Genetic Algorithms based optimization method working in finished precision, in a quantified research space, applied to the synthesis of digital filters with real coefficients coded in fixed-point arithmetic. The realized works of simulations show the efficiency and the importance of this method in terms of optimization and synthesis of digital filters in off-line mode.

REMERCIEMENTS

Nous remercions tout d'abord, DIEU le Tout Puissant qui par sa générosité, nous a donné la volonté et la patience durant toutes nos longues années de préparation. Nous le remercions aussi de nous avoir permis d'arriver à bout de ce mémoire.

Je tiens à exprimer ma profonde reconnaissance et mes chaleureux remerciements à mes encadreurs ; Mr GUESSOUM Abderrazak, et Mr BENOUARED Abdelhalim, pour avoir accepté de diriger ce mémoire, et de m'avoir guidé et orienté tout au long de la réalisation de ce travail en prodiguant leurs précieux conseils et leurs vifs encouragements, en m'ayant laissé la plus grande liberté dans la conduite de mes travaux et encouragé dans cette tentative toujours difficile qui consiste à explorer des approches nouvelles à la frontière entre des disciplines scientifiques distinctes.

Je remercie spécialement le président du jury : Monsieur Hassan SALHI qui a accepté d'en être Président et dont les commentaires sur mon travail sont autant d'encouragements à poursuivre mes efforts

Que soient également remercié Mr Zoubir BENSLAMA et Mr Mohamed DJENDI pour leurs collaborations et encouragements, ainsi que leurs remarques pertinentes, qui ont permis d'enrichir notre travail.

Enfin, on ne saurait oublier de remercier les Membres de ce jury qui ont eu l'amabilité d'examiner notre travail, ainsi à tous ceux qui ont contribué de prés ou de loin à ce modeste mémoire. Les derniers remerciements vont à mes Chers parents et à ma Famille car sans eux, rien de ce qui est entre vos mains aujourd'hui n'aurait été réalisé.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

L'acquisition de signaux physiques, quels qu'ils soient, est une opération souvent délicate. Un simple convertisseur analogique-numérique permet de transformer une tension ou un courant en une donnée numérique. Cela suffirait si les valeurs physiques n'étaient pas étroitement associées à une multitude de signaux, autres signaux de mesure, signaux parasites tels que le bruit électronique, les décharges électrostatiques, les fluctuations de la tension d'alimentation. Tous ces signaux se combinent, s'interfèrent et viennent "polluer" le signal utile. Afin de capturer correctement le signal recherché, un convertisseur analogique-numérique est toujours associé à des convertisseurs, à des amplificateurs ou à des filtres. Le rôle de chacun est de mettre en forme les signaux pour les rendre exploitables, les convertisseurs transforment des fréquences, des courants, des impulsions en tension, les amplificateurs adaptent l'amplitude de la tension à la plage d'entrée du convertisseur analogique-numérique. Le filtrage permet, quant à lui, d'extraire une partie de l'information liée à un signal, d'éliminer des fréquences parasites indésirables.

Il existe deux types de filtrage, analogique et numérique, néanmoins les développements considérables dans le domaine de la microélectronique conduisent à l'avènement de calculateurs, de processeurs et de microprocesseurs toujours plus performants. Ceci élargit considérablement les domaines d'application des réglages numériques et en particulier les filtres numériques. Ces derniers ont résolu beaucoup de problèmes posés par les filtres analogiques tels que les erreurs liées aux non-linéarités des amplificateurs opérationnels **[1]**, ou aux mauvaises tolérances des résistances, des capacités, etc. Néanmoins, Les filtres numériques ne sont pas pour autant parfaits, leurs résultats sont entachés par d'autres sources d'erreurs. L'une d'entre elles est liée à la conversion analogique-numérique, causée par la numérisation qui consiste à transformer un signal analogique en un signal numérique (binaire) après échantillonnage **[2][3]**, Lors de cette étape, on introduit une erreur appelée « erreur de quantification ». Une autre source de bruit provient de l'architecture des microprocesseurs à virgule fixe **[4]**. Prenons l'exemple d'un filtre ayant des coefficients valant 0.0001 et 0.9999 pour deux d'entre eux et implémenté sur DSP (Digital Signal Processor) à virgule fixe. Ce genre de DSP autorise pour valeurs de paramètres 0, 0.1, 0.2, 0.3, ... Il faut donc arrondir les coefficients de

l'algorithme aux valeurs possibles du DSP. Cette opération implique une erreur supplémentaire: c'est l'erreur de calcul. On retrouve le même phénomène avec les résultats des opérations de base (multiplication, addition, soustraction et division).

Comme la plupart de problèmes de conception, la réalisation de filtres numériques, nécessite une multitude de critères et spécifications de conception qui sont parfois contradictoires (réponse en fréquence, phase, stabilité, …), et pour trouver une conception optimale, la tache n'est pas du tout facile. Plusieurs méthodes ont été utilisées pour la conception des filtres **[5]**, qu'on peut classer en deux principales classes, les méthodes itératives et les méthodes d'optimisation.

Les méthodes itératives convergent généralement vers des conceptions sous-optimales, et par conséquent il était nécessaire de faire appel aux méthodes d'optimisation. Cependant, les problèmes d'optimisation formulés pour la conception de filtres numériques sont souvent complexes, idéalement la méthode d'optimisation doit convergé vers l'optimum global de la fonction objective le plus rapidement possible, les méthodes d'optimisation classiques sont généralement rapides et efficaces et ont donnée de bons résultats pour la conception de filtres numériques **[5]**, malheureusement ces méthodes sont bonnes pour l'obtention de minimum local mais elles ne sont pas équipées pour écarter les solutions inferieures en faveur des meilleures solutions.

Dans ces dernières années, une nouvelle variété d'algorithmes d'optimisation a été proposée, se basant sur le mécanisme de la sélection naturelle et la génétique, connue sous l'appellation « d'algorithmes génétiques (AG)». Ayant trouvé des solutions sous-optimales, les algorithmes «génétiques peuvent les écarter en faveur d'autres solutions sous-optimales plus prometteuses et par conséquent avec le temps, il ya plus de chance d'obtenir de meilleurs solutions pour des problèmes multicritères. Appliqués à la conception, ces algorithmes peuvent être programmés pour écarter les conceptions ayant des propriétés inferieures et des caractéristiques indésirables. Pour ces raisons, les algorithmes génétiques ont des chances d'atteindre les solutions optimales globales dans la conception de divers filtres numériques.

L’objectif de notre travail a été réalisé en deux phases, la première consistait à étudier les algorithmes génétiques et de considérer leurs approches à la conception de divers types de filtres numériques. Dans la seconde phase, on étudiait les effets des erreurs de quantification (arrondi, troncature) sur les performances des filtres numériques, et l'élaboration d'algorithme à base d'AG capable de faire la synthèse de filtres numériques , et d'élaborer des coefficients directement représentés en précision finie, autrement dit, les coefficients de la fonction de transfert du filtre obtenu, seront représentés en un nombre de bits limité (ex : 12 bits, 16 bits ou 20 bits) **[6]**.

Et par conséquent, ce mémoire sera structuré en cinq chapitres :

Dans le **premier chapitre**, on exposera d'une manière succincte, des rappels qui seront utilisés tout au long de ce travail. On présentera les caractéristiques des filtres numériques et en particulier les filtres récursifs, leurs types et les méthodes de conception utilisées. Une comparaison est établie entre les filtres numériques et analogiques d'une part, et entre les filtres numériques à réponse impulsionnelle infinie (RII) et les filtres numériques à réponse impulsionnelle finie (RIF) d'autre part.

Des notions sur la représentation des nombres et l'arithmétique binaire seront décrites au **chapitre 2**, ainsi que les différents types d'erreurs de quantification, et tout ce qui concerne leurs sources et leurs effets sur les systèmes linéaires tel que les filtres numériques.

Le **chapitre 3** nous introduira dans le domaine des métaheuristiques et en particulier les algorithmes génétiques, où on expliquera leur développement et l'importance des opérateurs génétiques: sélection des individus, croisement et mutation. Seront présentés aussi les différents types de codage des chromosomes porteurs d'informations (réel, binaire, ...) dans diverses processus d'algorithmes génétiques standards (AGS) et hiérarchiques (AGH).

Le **chapitre 4** présentera l'application des algorithmes génétiques standards (AGS) et les algorithmes génétiques hiérarchiques (AGH) à la conception de filtres numériques récursifs, on verra comment les algorithmes génétiques permettent de concevoir directement n'importe quel filtre sans passer par les transformations utilisés par les méthodes analytiques (transformations bilinéaires, invariance impulsionnelle, …) **[5].**

Seront définis aussi, les codages utilisés et surtout l'élaboration de la fonction d'évaluation (fitness) nécessaire pour le déroulement du processus génétique.

Enfin, le **dernier chapitre**, sera consacré à la présentation des résultats et aux interprétations des différentes simulations réalisées. Ces simulations concernent la synthèse de filtres RII en précision infinie par les algorithmes génétiques standards (AGS) puis par les algorithmes génétiques hiérarchiques (AGH), ensuite la synthèse de filtres numériques en précision finie. Une comparaison est faite entre les filtres obtenus et d'autres filtres classiques (filtres elliptiques par transformation bilinéaire).

CHAPITRE 1

LES FILTRES NUMERIQUES, SYNTHESE ET IMPLEMENTATION

1.1 Définition d'un filtre numérique

En électronique, un filtre numérique est un élément qui effectue un filtrage à l'aide d'une succession d'opérations mathématiques sur un signal discret. C'est-à-dire qu'il modifie le contenu spectral du signal d'entrée en atténuant ou éliminant certaines composantes spectrales non-désirées. Contrairement aux filtres analogiques, qui sont réalisés à l'aide d'un agencement de composants physiques (résistance, condensateur, inductance, transistor, etc.), les filtres numériques quant à eux, ils sont réalisés soit par programmation dans des circuits dédiés (processeurs de traitement du signal, DSP, µContrôleur, PC), ou sur des architectures programmables (FPGA, ASIC,…).

Les filtres numériques peuvent, en théorie, réaliser la totalité des effets de filtrage pouvant être définis par des fonctions mathématiques ou des algorithmes. Les deux principales limitations des filtres numériques sont la vitesse et le coût. La vitesse du filtre numérique est limitée par la vitesse (l'horloge, le "*clock*" en anglais) du processeur. Pour ce qui est du coût, celui ci dépend du type et des performances du processeur utilisé.

On appelle filtrage numérique de fréquences ou filtrage numérique linéaire toute combinaison linéaire des échantillons d'entrée et de sortie d'un système échantillonné (système numérique dans la quasi-totalité des cas) **[7]**. Si les signaux d'entrée ont été échantillonnés à la fréquence f_e (soit un pas d'échantillonnage ou période d'échantillonnage *Te)*, la relation linéaire la plus générale est **[5]**:

$$b_0 y(kT_e) = \sum_{m=0}^{M-1} a_m x(kT_e - mT_e) + \sum_{q=1}^{Q-1} b_q y(kT_e - qT_e) \quad (1.1)$$

Que l'on écrit le plus souvent :

$$b_0 y_k = \sum_{m=0}^{M-1} a_m x(k-m) + \sum_{q=1}^{Q-1} b_q y(k-q) \quad (1.2)$$

Avec: x: Signal d'entrée, y: Signal de sortie, b_0, a_i, b_i: coefficients du filtre.

D'une façon générale, pour simplifier, on prend $b_0 = 1$.

En passant au domaine fréquentiel, moyennant la transformée en Z, on déduit de l'équation (1.1), l'expression de la fonction de transfert, qui elle aussi permet de définir un filtre numérique.

Ainsi, la fonction de transfert générale d'ordre Q d'un filtre numérique est la suivante:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{1+\sum_{q=1}^{Q-1} a_q z^{-q}} \quad (1.3)$$

Rappelons que la réponse en fréquence s'obtient en remplaçant **z** par :

$$\exp(\pi j f) = \cos \pi f_n + j \sin \pi f_n \quad (1.4)$$

Avec $f_n = 2\,(f / f_e)$, f_e *:* fréquence d'échantillonnage.

La relation (1.3) montre que la fonction de transfert d'un filtre numérique linéaire a obligatoirement la forme d'un rapport de deux polynômes algébriques en *z*, les valeurs des coefficients a_k et b_k fixent le type du filtre (passe-bas, passe-haut, etc.…).

1.2 Différents types de filtres numériques

La relation (1.3) représente la fonction de transfert en *z* la plus générale pour un filtre linéaire. Un tel filtre est appelé ***récursif,*** parce qu'en vertu de la relation (1.2), le calcul d'une valeur y_k de la sortie nécessite non seulement les valeurs antérieures de l'entrée, mais aussi les valeurs antérieures de la sortie **[8][9]**. Le schéma le plus général de ce filtre est représenté par la figure 1.1.

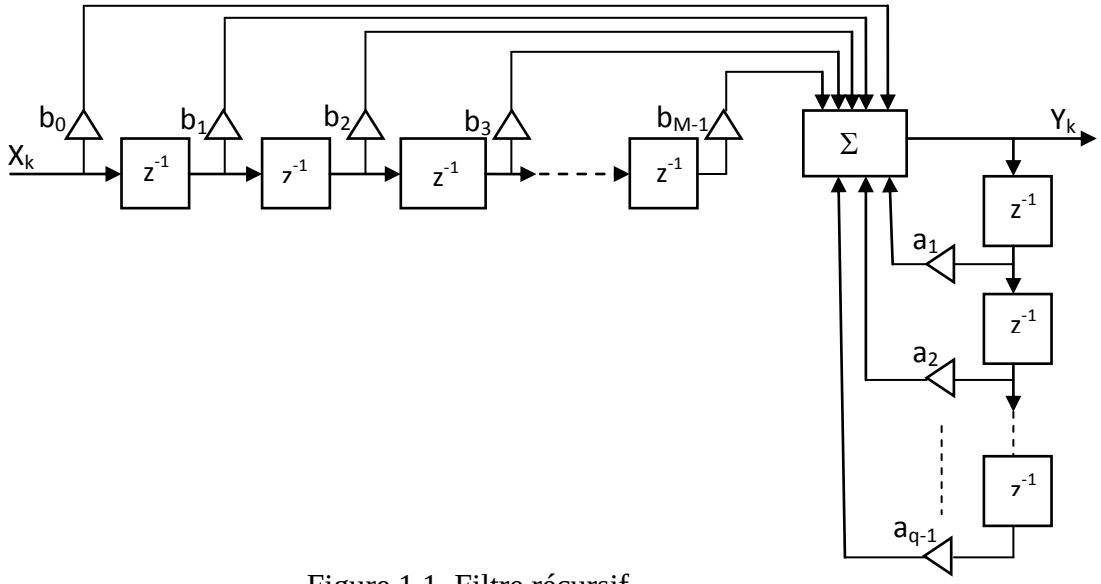


Figure 1.1 Filtre récursif

1.2.1 Filtre autorégressif

Ce cas général se simplifie si tous les b_m de la relation (1.3) sont nuls sauf b_0. La fonction de transfert en **z** est alors, d'après la relation (1.3) **[9]**:

$$H(z) = \frac{b_0}{1+\sum_{q=1}^{Q-1} a_q z^{-q}} \tag{1.5}$$

D'où $Y(z) = b_0 X(z) + \sum_{q=1}^{Q-1} a_q Y(z) z^{-q}$

Ou encore

$$Y(z) = b_0 X(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) + \cdots + a_q z^{-q} Y(z) + \cdots + a_{q-1} z^{q-1} Y(z)$$

Ou, en passant dans le domaine temporel :

$$Y_k = b_0 X_k + a_1 Y_{(k-1)} + a_1 Y_{(k-2)} + \cdots + a_q Y_{(k-q)} + a_{Q-1} Y_{(k-Q+1)} \tag{1.6}$$

La relation (1.6) montre que la sortie Y_k ne dépend pas des M valeurs précédentes de x, mais seulement de la valeur actuelle de x et des Q valeurs des échantillons des sorties précédentes. Un tel filtre est appelé **autorégressif** (AR). Son schéma est représenté par la figure (1.2).

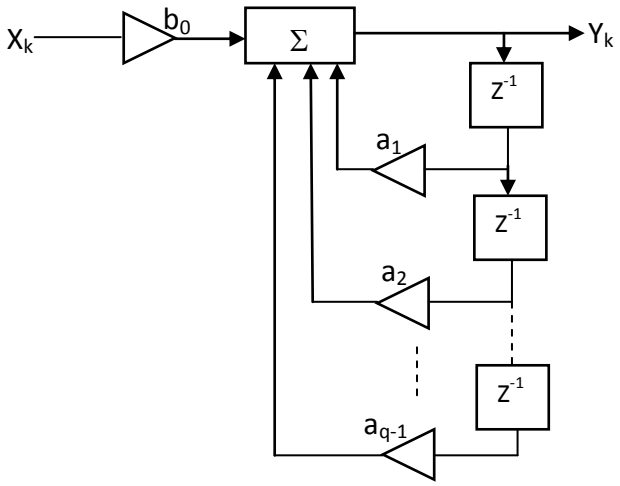


Figure 1.2 Filtre autorécursif

1.2.2 Filtre transverse (ou encore moyenne adaptée MA)

Le cas général se simplifie aussi si tous les coefficients a_i de la relation (1.3), sont nuls sauf a_0 que l'on prend égal 1 pour simplifier, la fonction de transfert en z devient alors :

$$H(z) = \sum_{m=0}^{M-1} b_m z^{-m}$$

La fonction de transfert ne comporte plus de dénominateur et l'on a:

$$Y(z) = \sum_{m=0}^{M-1} b_m X(z) z^{-m}$$

Ou, en revenant dans le domaine temporel:

$$y_k = \sum_{m=0}^{M-1} b_m x(k-m) \qquad (1.7)$$

La sortie y_k ne dépend que des entrées précédentes. Un tel **filtre** est appelé **transverse** ou **transversal** (sans doute à cause de la forme de son schéma représenté par la figure 1.3), ou encore **filtre non récursif** ou **filtre MA** (moyenne adaptée en français, *moving average* en anglais) **[10]**.

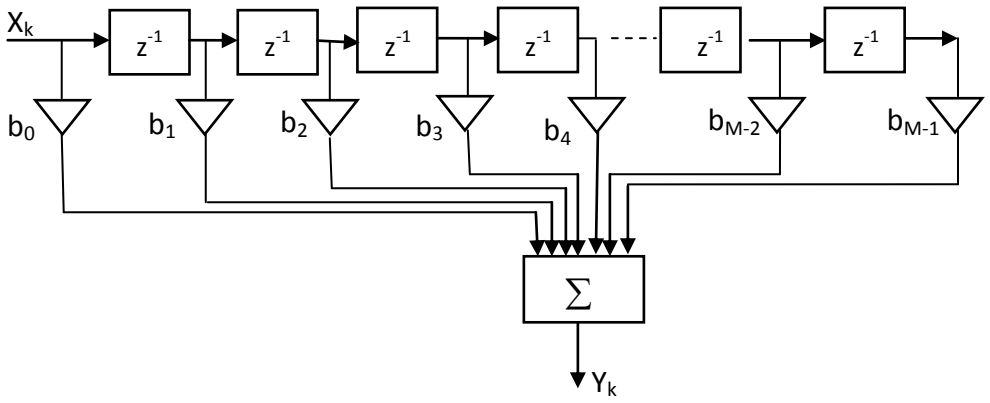


Figure 1.3 Filtre transverse (ou filtre à moyenne adaptée)

La relation (1.7) n'est autre que l'équation de convolution numérique dans le domaine temporel ; les coefficients b_m sont les valeurs successives de la réponse impulsionnelle du filtre numérique, la réponse impulsionnelle est donnée par :

$$h_M(m) = \sum_{m=0}^{M-1} h(mT_e)\delta(t - mT_e)$$

Du fait que le nombre de termes de la somme de la relation est limité, les filtres non récursifs sont souvent appelés **filtres à réponse impulsionnelle finie** (de durée finie), **filtres RIF** (ou FIR en anglais).

Par contre, si nous considérons une fonction de transfert en *z*, récursive, équation (1.3), et si l'on opère la division du numérateur par le polynôme algébrique en *z* du dénominateur, on trouvera, en général, une infinité de termes; un filtre récursif est, pour cette raison, appelé parfois **filtre à réponse impulsionnelle infinie** (de durée infinie), **filtre RII** (ou IIR en anglais).

Le filtre le plus complet peut être considéré comme l'association en cascade d'une fonction de transfert autorégressive et d'une fonction de transfert à moyenne adaptée, d'où la dénomination **ARMA** (*autorégressive and moving average*) parfois donnée au filtre numérique le plus général.

1.3 Stabilité des filtres numériques

On dira qu'un filtre numérique est stable, si à toute entrée x_n bornée par la valeur M correspond une sortie y_n bornée par une valeur M' **[11]**:

$$\exists M \ \forall n \ |x_n| < M \qquad \Rightarrow \qquad \exists M' \ \forall n \ |y_n| < M'$$

Il existe par conséquent, deux conditions nécessaires et suffisantes, pour qu'un filtre numérique soit stable :

- **1ére condition nécessaire et suffisante de stabilité**

Les échantillons de la réponse impulsionnelle sont bornés par une valeur A, et cette condition s'écrit:

$$\sum_{n=-\infty}^{+\infty} |h(n)| < A$$

- **2éme condition nécessaire et suffisante de stabilité**

Lorsque le filtre est causal (paragraphe 1.4.2) et que sa fonction de transfert est rationnelle, il existe une autre condition nécessaire et suffisante de stabilité: il faut et il suffit que tous les pôles de H(z) soient à l'intérieur du cercle unité du plan z **[11]**.

1.4 Filtre à réponse impulsionnelle finie (RIF)

Les filtres numériques à réponse impulsionnelle finie (RIF) sont des systèmes linéaires et invariants dans le temps définis par une équation selon laquelle un nombre de sortie, représentant un échantillon du signal filtré, est obtenu par sommation pondérée d'un ensemble fini de nombre d'entrée, représentant les échantillons du signal à filtrer. Les coefficients de la sommation pondérée constituent la réponse impulsionnelle du filtre et un ensemble fini d'entre eux seulement prennent des valeurs non nulles.

En quelque sorte, le système oublie les valeurs les plus anciennes au profit des plus récentes, il est à mémoire finie. Cette mémoire étant constituée par les coefficients de pondération.

La suite d'entrée $x(n)$ et la suite de sortie $y(n)$ sont reliées par l'équation suivante :

$$y(n) = \sum_{i=0}^{N-1} b_i \, . \, x(n-i) \tag{1.8}$$

Le filtre ainsi défini comporte un nombre N (fini) de coefficients b_i considéré comme un système discret, il a pour réponse impulsionnelle, la suite $h(i)$ définie de la manière suivante :

$$h(i) = \begin{cases} b_i & 0 \leq i \leq N-1 \\ 0 & ailleurs \end{cases} \tag{1.9}$$

C'est-à-dire que la réponse impulsionnelle est tout simplement la suite des coefficients b_i
Une telle structure est représentée à la figure 1.4 :

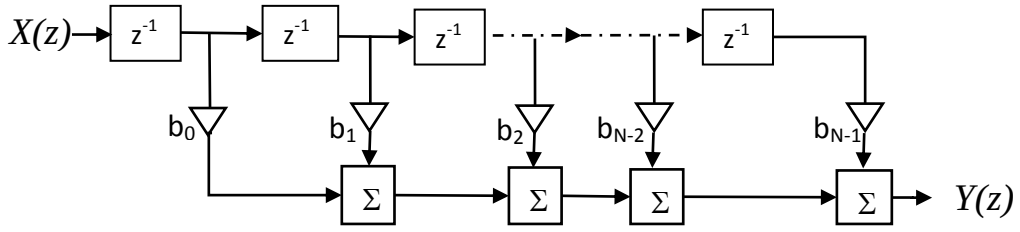


Figure 1.4 Structure d'un filtre RIF

1.4.1 Stabilité des filtres RIF

La fonction de transfert H(z) d'un filtre RIF ne possède pas de pôles (uniquement des zéros), par conséquent et d'après le paragraphe §1.3, le filtre RIF est inconditionnellement stable.

1.4.2 Fonctions de transfert réalisables

Un filtre numérique traitant des nombres représentant les échantillons du signal prélevés par période T_e ($T_e = 1/f_e$, f_e : fréquence d'échantillonnage), possède une réponse en fréquence périodique. Par suite, en peut développer sa réponse en fréquence en série de Fourier comme suite :

$$H(f) = \sum_{n=-\infty}^{+\infty} C_n e^{j2\pi\frac{nf}{f_e}} \quad (1.10)$$

Avec $$C_n = \frac{1}{f_e}\int_0^{f_e} H(f).e^{-j2\pi n\frac{f}{f_e}}\ df \quad (1.11)$$

De même, La fonction *H (f)* peut être approchée par un développement réduit à un nombre limité de termes :

$$H(f) = \sum_{n=-q}^{p} C_n e^{j2\pi\frac{nf}{f_e}} = H_L(f) \quad (1.12)$$

Où p et q sont deux entiers finis ; l'approximation est d'autant meilleur que ces nombres sont plus grands.

La propriété de causalité, qui traduit le fait que dans un filtre réel la sortie ne peut précéder l'entrée dans le temps, implique que la réponse impulsionnelle $h(n)$ soit nulle pour $n < 0$. Donc si le filtre est causal, alors q=0, on obtient :

$$H_L(f) = \sum_{n=0}^{p} C_n e^{j2\pi\frac{nf}{f_e}} \quad (1.13)$$

Il est résulte que toute fonction de filtrage numérique causale peut être approchée par la fonction de transfert d'un filtre RIF.

1.4.3 Synthèse des filtres RIF

Le problème consiste à trouver les coefficients b_i qui constituent les mémoires du système, ces coefficients doivent être calculés pour que le filtre satisfasse des contraintes imposées, ces contraintes peuvent être définies dans l'espace du temps, par exemple, par la donnée de la réponse impulsionnelle, elles sont cependant, le plus souvent, définie dans l'espace des fréquences, la fonction de transfert devant approcher une fonction désirée et rester à l'intérieur d'un gabarit qui constitue les spécifications.

Pour un filtre passe-bas, on impose par exemple à la réponse en amplitude d'approcher la valeur 1 avec une précision, dans la bande de fréquence [0 , f_p] dite

« *bande passante* », et la valeur 0 avec une précision δ_2, dans la bande [f_a , f_e /2] dite « *bande atténuée* ». Le gabarit Correspondant est représenté par la figure (1.5) :

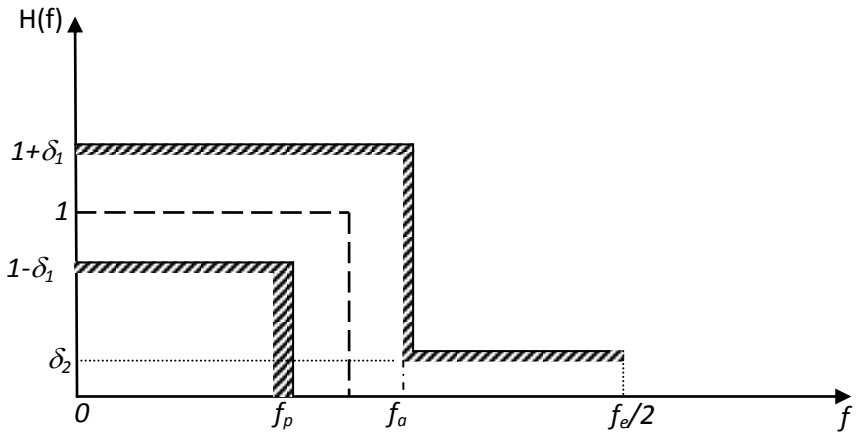


Figure 1.5 : Gabarit d'un filtre passe-bas.

L'intervalle $f \in [f_a, f_p]$ est appelé *Bande de transition* et *la raideur de coupure* est désigné par le paramètre R_c tel que :

$$R_c = \frac{f_p + f_a}{2(f_a - f_p)}$$

Pour calculer les coefficients de filtre RIF, on peut utiliser plusieurs méthodes, ces méthodes peuvent être résumés par l'organigramme suivant:

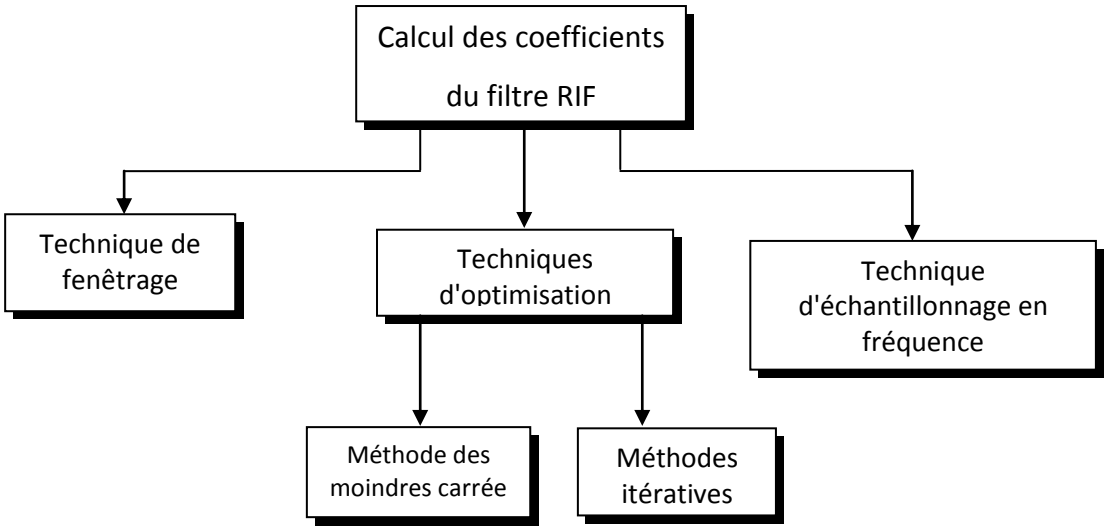


Figure 1.6 : Techniques de calcul des coefficients de filtre RIF

Parmi les techniques citées par l'organigramme de la figure (1.6), la méthode optimale nommée aussi méthode minimax, a été très utilisée par l'appel de certain algorithmes tels Parks-Maclellan et Remez **[7]**.

Cette méthode est basée sur une distribution uniforme de l'ondulation sur l'ensemble de la bande passante et de l'affaiblissement sur toute la bande de réjection. Les filtres FIR résultants ou filtres "elliptique" sont nettement plus performants que les filtres FIR fenêtrés, et leurs réponses en phase est également linéaire. Cette méthode recherche itérativement les paramètres pour que le gabarit soit respecté au mieux, avec un ordre du filtre minimal. Par rapport à un filtre FIR fenêtré, l'ordre d'un filtre FIR " elliptique " est nettement inférieur (à gabarit identique), l'ondulation dans la bande passante et l'affaiblissement minimal dans la bande de réjection sont configurables séparément. Il existe une autre méthode basée l'un algorithme des moindres carrés. Par rapport à l'algorithme Parks-Maclellan, elle présente une ondulation dans la bande passante moindre et un affaiblissement minimal dans la bande de réjection plus accentué. La transition bande passante / bande de réjection est cependant bien moins rapide.

En règle générale, on procède en quatre étapes :

1- Résolution du problème d'approximation pour déterminer les coefficients du filtre qui satisfait un gabarit en fréquence donné ;
2- Choix d'une structure et quantification des coefficients du filtre en un nombre fini de bits ;
3- Quantification des variables du filtre, c'est-à-dire le choix d'une longueur de mots pour :
    - ✓ l'entrée,
    - ✓ la sortie,
    - ✓ les mémoires intermédiaires;

4- Vérification par simulation si le filtre final satisfait le gabarit fixé.

Nous nous intéressons ici à la première étape. L'étape 4 n'est nécessaire que si les étapes 2 et 3 sont mises en œuvre; or le choix d'une structure de réalisation et les problèmes de quantifications ne sont pas indépendants de l'architecture du calculateur utilisé pour la réalisation pratique. De plus, il faudrait tenir compte de la notion d'efficacité

de calcul (algorithmique rapide) et de manière paradoxale, ce n'est pas toujours l'algorithme qui nécessite le moins d'opérations qui s'avère le plus efficace pour une architecture donnée. L'ensemble de ses problèmes que l'on peut réunir sous la notion d'adéquation architecture-algorithme, ne possède pas de réponse unique et leur étude pourrait faire l'objet d'un développement plus complet.
Parmi les avantages des filtres RIF, on peut citer :

- ✓ la possibilité de réaliser des filtres à phase linéaire ;
- ✓ la possibilité d'obtenir un bruit de calcul assez faible.

Parmi les inconvénients, il y a :

- ✘ la nécessité d'un ordre assez élevé pour obtenir des filtres à coupure raide ;
- ✘ pour les filtres à phase linéaire, un temps de propagation de groupe n'est pas forcément un nombre entier d'échantillons.

1.5 Filtres à réponse impulsionnelle infinie

Les filtres numériques à réponse impulsionnelle infinie sont des systèmes linéaires discrets invariants dans le temps dont le fonctionnement et régi par une équation de convolution portant sur une infinité de termes. En principe, ils conservent une trace des signaux qui leur ont été appliqués pendant une durée infinie : ils sont à mémoire infinie. Une telle mémoire est réalisée par une boucle de réaction de la sortie sur l'entrée, d'où la dénomination courante de ***filtre récursif***.

Les filtres récursifs présentent des avantages qui sont :

- ✓ une bande de transition étroite pour un ordre équivalent à celui d'un filtre non récursif ;
- ✓ la possibilité de réaliser des déphaseurs purs dits aussi filtre passe-tout.

Les méthodes de synthèse de ces filtres se classent en deux catégories :

- **Les méthodes de transposition** : elles partent du principe que le problème de la synthèse des filtres a déjà été largement développé dans le domaine du signal continu. Il a été établi plusieurs méthodes d'approximation polynômiale d'un gabarit aboutissant aux filtres de Butterworth, Tchebychev, elliptique(Cauer), Legendre…. Le filtrage des signaux discrets reprend ces méthodes en cherchant simplement une technique de passage du continu au discret qui préserverai au mieux les caractéristiques du filtre.

- **Les méthodes directes** : elles cherchent à faire une synthèse directe dans le domaine discret à partir d'un gabarit. Ce sont des méthodes itératives basées sur la minimisation d'un critère comme celui des moindres carrés utilisé par l'algorithme de Fletcher et Powel.

Pour aborder l'étude des filtres RII, il est plus simple de considérer d'abord les cellules de filtres élémentaires du premier et du second ordre. En fait, l'intérêt de ces structures simples va bien au-delà d'une introduction aux propriétés des filtres RII, car en pratique, les filtres RII même les plus complexes se présentent sous la forme d'une combinaison d'un ensemble de ces cellules élémentaires.

1.5.1 Fonction de transfert

La relation entrée-sortie d'un filtre récursif est donnée par la relation :

$$y(n) = \sum_{i=0}^{q} b_i\, x(n-i) - \sum_{j=0}^{p} a_i\, y(n-j) \qquad (1.15)$$

Si on passe à la transformée en Z de l'équation (2.15), on obtient la fonction rationnelle :

$$H(z) = \frac{\sum_{i=0}^{q} b_i z^{-i}}{1+\sum_{j=1}^{p} a_j z^{-j}} \qquad (1.16)$$

Avec $p \geq q$ (souvent, en prend $p = q = N$, N étant l'ordre du filtre).

1.5.2 Structures de réalisation d'un filtre RII

Pour être réalisé, un filtre numérique doit être donné sous forme de structure. Il existe plusieurs types de réalisations citées ci-dessous :

1.5.2.1 Structure directe:

On peut distinguer deux formes différentes:

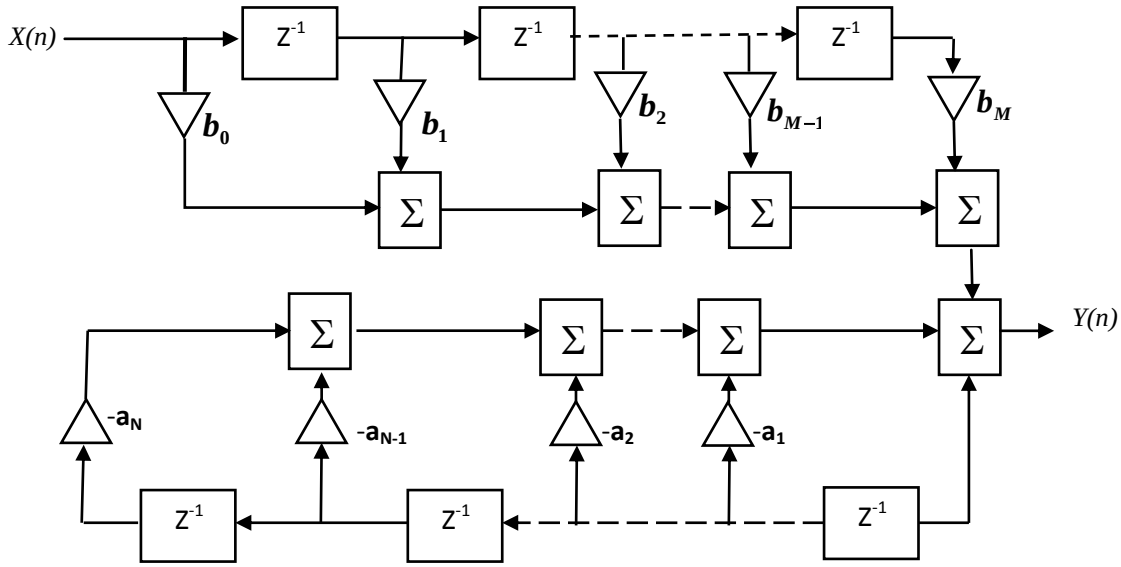


Figure 1.7.a : Structure directe type I d'un filtre RII

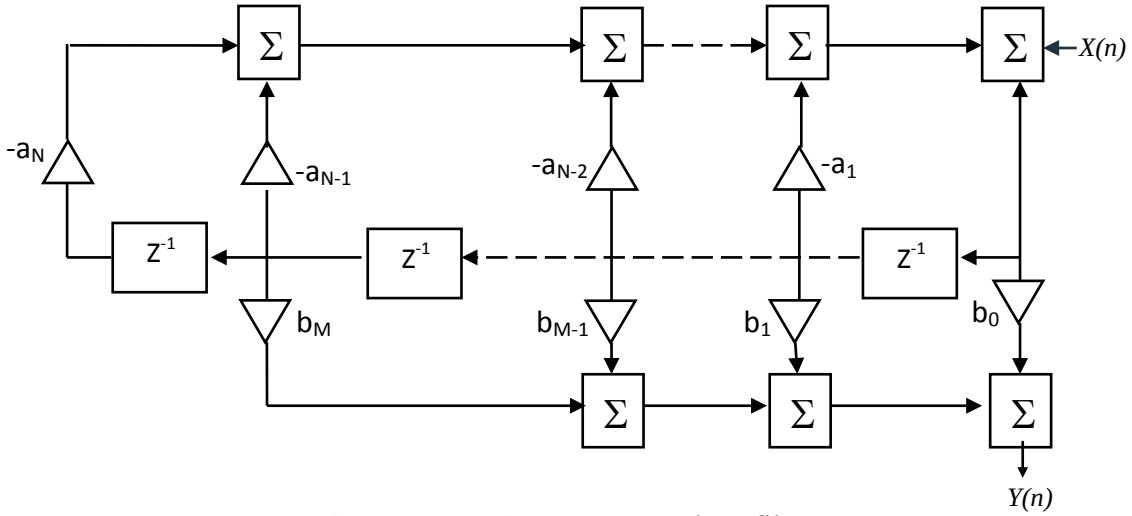


Figure 1.7.b : Structure directe type II d'un filtre RII

$$H(z) = \frac{Y(z)}{R(z)} . \frac{R(z)}{X(z)} = H_1(z).H_2(z) \tag{1.17}$$

$$H_1(z) = \sum_{i=0}^{N} b_i . z^{-i} \quad et \quad H_2(z) = \frac{1}{1 + \sum_{j=1}^{N} a_j . z^{-j}} \tag{1.18}$$

$$H_2(z) = \sum_{i=0}^{N} b_i . z^{-i} \quad et \quad H_1(z) = \frac{1}{1 + \sum_{j=1}^{N} a_j . z^{-j}} \tag{1.19}$$

1.5.2.2 Structure décomposée

En pratique, la structure directe est peu utilisée, donc au lieu de réaliser $H(z)$ directement, on effectue une décomposition en somme (structure parallèle) ou en produit (structure cascade) des fonctions élémentaires du premier ou de second ordre réalisées séparément.

On peut représenter les deux formes comme suite :

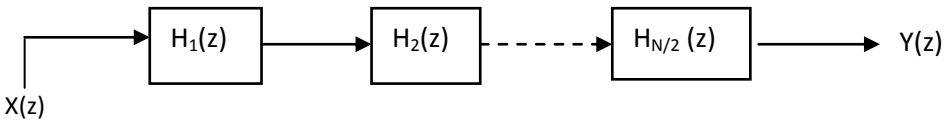


Figure 1.7.c : Structure en cascade

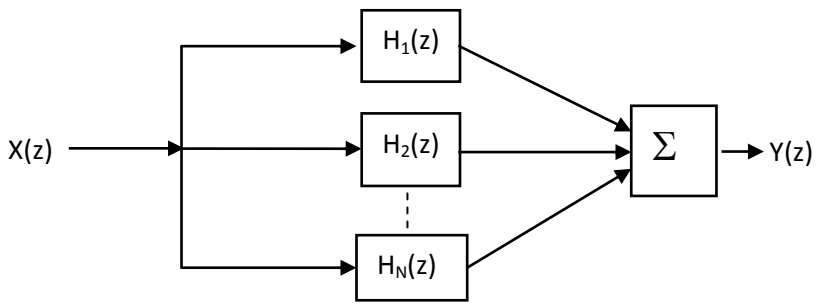


Figure 1.7.d : Structure en parallèle

1.5.2.3 Cellule du premier ordre

La réponse impulsionnelle d'une cellule du 1er ordre est donnée par la relation de récurrence suivante :

$$y(n) = x(n) - a.y(n-1) \qquad \text{Avec} \quad a \in \Re \tag{1.20}$$

Le circuit comprend une mémoire des données et une mémoire des coefficients. Il faut pour chaque nombre de sortie effectuer une multiplication et une addition.

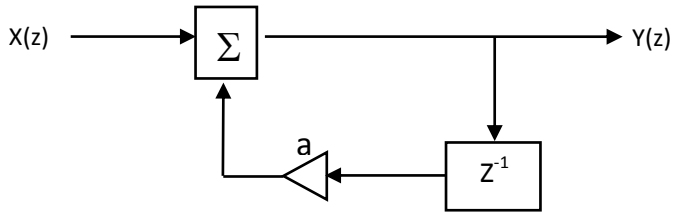


Figure 1.8 : Structure d'une cellule du premier ordre

A partir de la relation de récurrence (1.20) on tire :

$$H(z) = \frac{y(z)}{x(z)} = \frac{1}{1 + a.z^{-1}} = \frac{z}{z + a} \tag{1.21}$$

Pour que la fonction $H(z)$ soit stable, il faut que tous les pôles se trouvent à l'intérieur de cercle unité, c'est-à-dire $|a| < 1$.

Dans cette analyse, il résulte que la cellule du premier ordre offre des possibilités restreintes car elle ne possède qu'un seul pôle, qui doit être réel pour que le filtre soit à coefficient réels, et sa réponse en fréquence une fonction monotone.

Par contre, la cellule de second ordre offre des possibilités beaucoup plus variées. C'est la structure la plus utilisée en filtrage numérique en raison de la modularité qu'elle apporte dans la réalisation des filtres même les plus complexes.

1.5.2.4 Cellule du second ordre

Soit un système qui à la suite de donnée $x(n)$ fait correspondre la suite $y(n)$ telle que :

$$y(n) = x(n) - a_1.y(n-1) - a_2.y(n-2) \tag{1.22}$$

Cette structure correspond à un circuit à deux mémoires de données et une mémoire de coefficients.

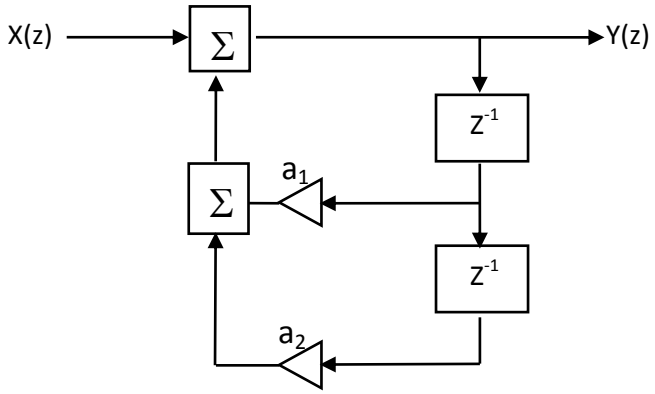


Figure 1.9 : Structure d'une cellule du second ordre

Pour chaque échantillon de sortie, il faut deux additions et deux multiplications.

La fonction de transfert en z, H(z) du système, est donnée par :

$$H(z) = \frac{1}{1 + a_1.z^{-1} + a_2 z^{-2}} = \frac{z^2}{z^2 + a_1 z + a_2} \tag{1.23}$$

Cette fonction possède, suivant le signe de $a_1{}^2 - 4a_2$ des pôles réels ou complexes conjugués :

Si $a_1{}^2 \geq 4.a_2$: on a deux pôles réels, leurs images, sont situés sur l'axe des réels. La fonction de transfert est alors le produit de deux fonctions du premier ordre, ces fonctions sont identiques si $a_1{}^2 = 4.a_2$.

Si $a_1{}^2 < 4.a_2$: les deux pôles sont complexes conjugués. Leurs images sont *p* et *p** tel que :

$$z_p = \frac{1}{2}\left(-a_1 + j\sqrt{4.a_2 - a_1{}^2}\right) \tag{1.24}$$

1.5.3 Stabilité des filtres IIR

Pour que le filtre soit stable, il faut que le module de ses pôles soit inférieur à 1.

1.5.3.1 Cas des pôles réels

$$\frac{1}{2}\left(-a_1 + \sqrt{a_1{}^2 - 4.a_2}\right) < 1 \quad et \quad \frac{1}{2}\left(-a_1 - \sqrt{a_1{}^2 - 4.a_2}\right) > -1 \tag{1.25}$$

D'où : $$|a_1| < 1 + a_2 \tag{1.26}$$

1.5.3.2 Cas des pôles complexes

Pour qu'un filtre à pôles complexes soit stable il faut que : $\Rightarrow$ $a_2 < 1$

Le domaine de stabilité est donc un triangle délimité par trois droites d'équations :

$$a_2 = 1 \quad a_2 - 1 - a_1 \quad a_2 = -1 + a_1$$

De plus, le domaine des pôles réels est séparé de celui des pôles complexes par la parabole d'équation: $a_1{}^2 = 4.a_2$.

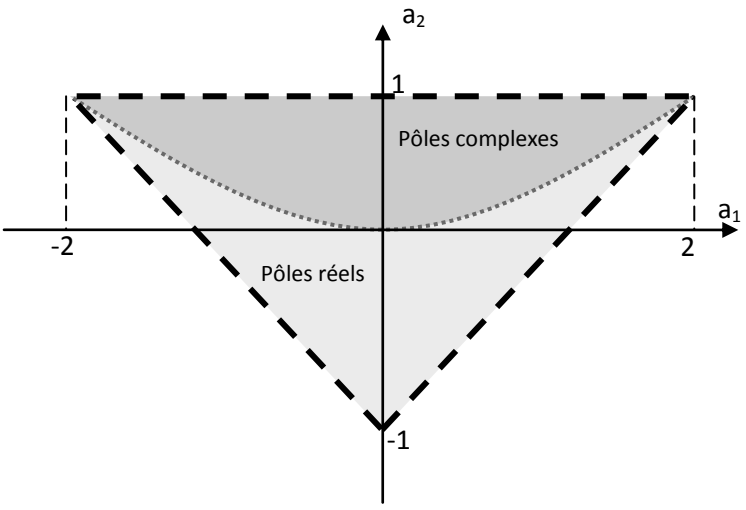


Figure 1.10 : Triangle de stabilité de filtre récursif

1.5.4 Cellule du second ordre générale

La cellule du second ordre la plus générale, fait intervenir dans le calcul d'un élément de la suite de sortie $y(n)$ à l'instant *n*, les données aux instants précédents $x(n-1) \quad et \quad x(n-2)$, on l'a définie par l'équation suivante :

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2) \tag{1.27}$$

Il en résulte la fonction de transfert en z suivante :

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{1.28}$$

Cette expression correspond à la structure ci-dessous :

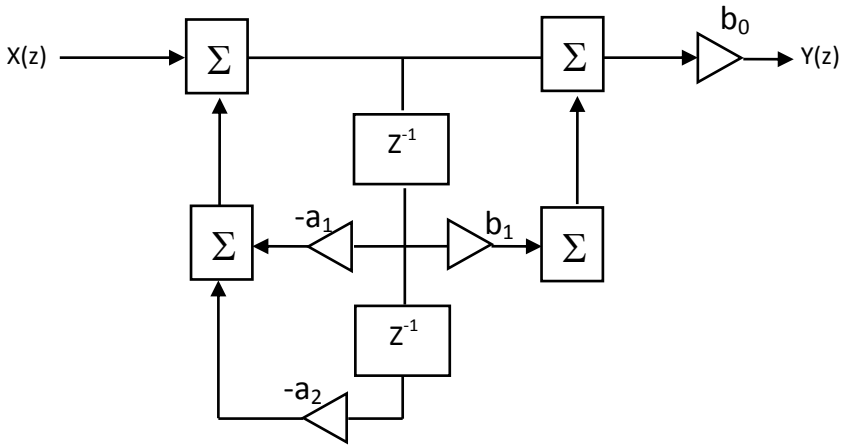


Figure 1.11 : Structure générale d'une cellule récursive du second ordre

Cette fonction comporte deux zéros et deux pôles. On peut la considérer comme le produit de deux fonctions :

$$H(z) = \left(b_0 + b_1 z^{-1} + b_2 z^{-2}\right)\left(\frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}\right) = \frac{N(z)}{D(z)} \tag{1.29}$$

Ainsi, le filtre considéré est la mise en cascade d'une cellule RII purement récursive et une cellule RIF. L'étude de la stabilité du filtre se ramène à celle de la cellule purement récursive qui la constitue.

1.6 Comparaison entre filtres numériques et filtres analogiques

Les avantages du filtrage numérique par rapport au filtrage analogique sont importants, on peut citer :

- **Reproductibilité :** Les caractéristiques de tous les filtres numériques établis sur une même configuration sont rigoureusement identiques,
- **Souplesse :** La réponse en fréquence peut être très aisément modifiée en changeant les coefficients arithmétiques ; le domaine de fréquences de travail est facilement déplacé par modification de la fréquence d'échantillonnage,
- **Précision :** Les différentes manipulations étant effectuées sur des nombres, la précision dépend, en grande partie, de celle du CAN et de celle du CNA,
- **Association des filtres:** La mise en série de filtres numériques ne pose aucun problème d'interaction, tel que celui que l'on rencontre pour l'adaptation des impédances des filtres analogiques,
- **Stabilité des caractéristiques :** Il n'ya pas de vieillissement des composants du à l'influence de la température sur les caractéristiques du filtre.

Les principaux inconvénients sont liés au problème de l'échantillonnage (spectre du signal toujours limité) nécessitant l'utilisation de processeurs ayant une bonne rapidité d'exécution pour pouvoir traiter des signaux ayant une forte « dynamique » (fréquences élevées) en temps réel.

1.7 Caractéristiques des filtres numériques

Généralement les spécifications du filtre numérique sont données à priori, mais dans certains cas le concepteur doit établir ces propres spécifications, ceci est le plus important pré-requis pour la conception d'un filtre numérique ; une fois établies, si les spécifications sont trop strictes (ex. tolérances sur la bande passante et sur la bande coupée trop faible, et une bande de transition trop mince), le filtre peut ne pas être réalisable (faisable).

La sélection propre des spécifications doit être faite en fonction de la nature des signaux (ex. bandes de fréquence et les niveaux correspondant de signaux ou bruits désirés ou non-

désirés ; et le software et le hardware disponible (arithmétique à virgule flottante ou fixe, longueur de mot (*wordlength*) des coefficients, etc.).
Comme la plupart des problèmes de conception, la réalisation de filtres numériques commence par la spécification explicite des exigences de performances. Ceci doit inclure :

- ✓ Les caractéristiques du signal d'entrée,
- ✓ Les caractéristiques de la réponse en fréquence du filtre,
- ✓ La manière d'implémentation (structure, software, hardware, …)
- ✓ Autres contraintes de conception (tel que le coût et la dégradation possible du signal).

En général, un filtre est défini par son gabarit. Ce gabarit est constitué des limites de tolérance pour les différents éléments du filtre, à savoir :

- ✓ La fréquence de coupure, f_c
- ✓ L'atténuation dans la bande coupée, en dB, Att_{dB}
- ✓ L'ondulation dans la bande passante, en dB, Δ_{ond}
- ✓ La largeur de la bande de transition, située entre la fréquence de coupure et la zone atténuée, car la coupure d'un filtre n'est jamais parfaite, Δ_{ftrans}

Il peut y avoir plusieurs fréquences de coupures dans le cas de filtres autres que passe-haut ou passe-bas. La figure ci-dessous illustre les différentes notations de gabarit suivant le type du filtre, avec f_p et f_s représentent respectivement la fréquence de la bande passante et la fréquence de la bande coupée.

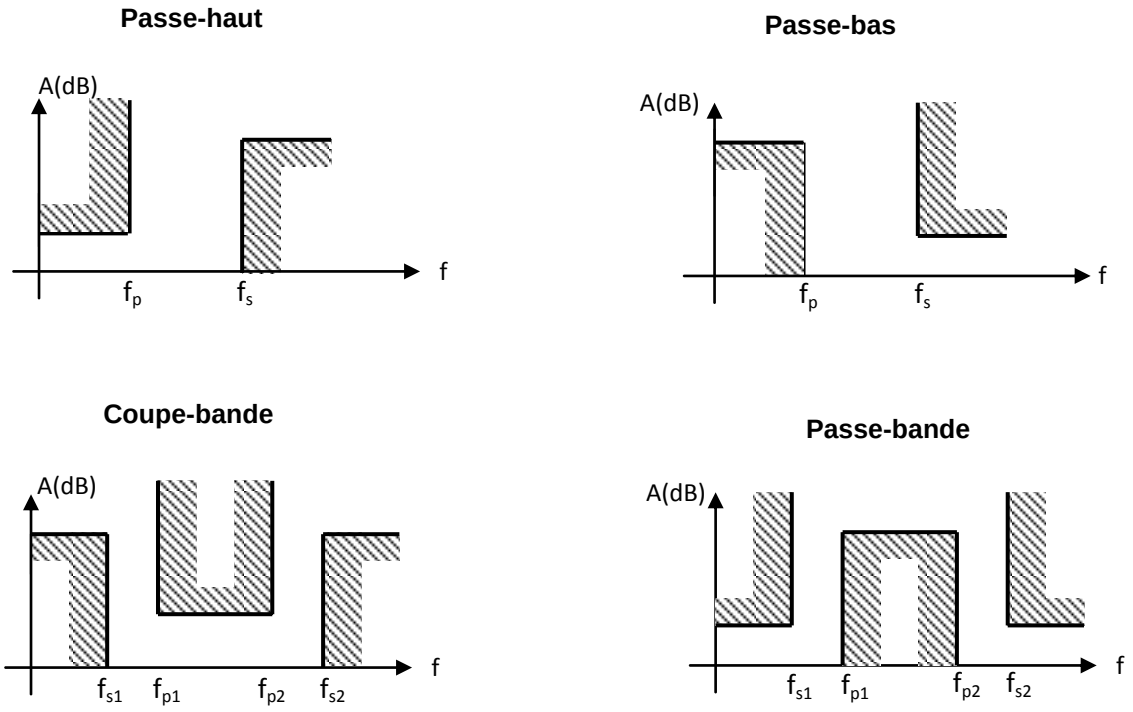


Figure 1.12 Différents gabarits de filtres numériques

1.7 Conclusion

Apparu dés que la puissance des ordinateurs et les performances des convertisseurs analogique-numérique l'ont permis, le filtrage numérique est devenu un outil fondamental dans tous les domaines ou l'on traite le signal.

Les méthodes décrites dans cet article permettent de construire de tels filtres pour des cas simples. Il ne faut pas oublier que le filtrage numérique impose une numérisation préalable du signal issu du capteur, et si les performances exigées sont modestes, cette opération n'est peut être pas justifiée. Un filtre analogique toujours stable et peu couteux peut suffire. Par contre, un filtre numérique peut avoir des performances hors de portée de tout filtrage analogique, si complexe soit-il. Des techniques sophistiquées sont souvent nécessaire **[5]**, et l'amélioration de techniques numériques de traitement du signal est un sujet de recherche très actuel qui fait l'objet de nombreuses publications de haut niveau.

CHAPITRE 2

ERREURS DE QUANTIFICATION ET LONGUEUR DE MOT FINIE

2.1 Introduction

Le calcul et la réalisation d'un filtre numérique supposent une grande précision (précision infinie), mais les grandeurs qui interviennent lors de leur implémentation sont codées par des mots de longueur finie. Les erreurs dues à la quantification sont donc inévitables et leurs conséquences dépendent fortement de plusieurs paramètres du filtre.

Les modes de représentation en virgule fixe et virgule flottante sont généralement supportées par les processeurs standards. La représentation en virgule fixe est la plus utilisée à cause de la grande vitesse d'exécution qu'elle offre avec relativement une unité arithmétique simple. Pour des exigences de vitesse, de coût et de confinement, son choix s'impose **[6]**.

Dans le chapitre précédent, nous avons supposé à chaque instant que tous les signaux, ainsi que d'autres grandeurs (comme les coefficients des filtres) pouvaient prendre n'importe quelle valeur. Pour la catégorie des signaux et des systèmes numériques, cette supposition ne tient pas, parce que chaque quantité est représentée par la combinaison d'un nombre fini de bits (mot binaire). Un bit est un nombre qui peut prendre seulement deux valeurs différentes (0 et 1). Avec une longueur de mot de B bits, on peut donc distinguer au plus 2^B valeurs différentes. Quand on est libre de choisir B, on peut rendre la représentation numérique aussi précise qu'on le veut et donc approcher tout signal numérique ou tout système numérique avec une précision aussi grande qu'on le souhaite.

Cependant, la pratique réelle est entièrement différente. Pour des raisons d'économie, nous sommes souvent intéressés de savoir comment on peut choisir la valeur de B la plus basse possible sans introduire d'erreurs inacceptables. Mais nous sommes alors inévitablement confrontés à un certain nombre d'effets de nature très variée, causés par la longueur de mot finie que nous utilisons. Ces effets sont souvent très compliqués et les seules conclusions que l'on peut tirer à leur sujet sont des conclusions statistiques.

Ceci est en partie dû à l'introduction de non-linéarités dans le système, ce qui rend toute description exacte du système compliquée, sinon impossible **[12]**.

Le filtrage numérique parait simple quand on travaille avec un logiciel tel que MATLAB (MATrix LABoratory) en double précision **[13]**. Les choses se compliquent lorsque les calculs doivent se faire en virgule fixe et à précision finie, et il ya des erreurs de quantification qui apparaissent lors de l'implémentation du filtre numérique et qui sont dues essentiellement à: **[14][15]**

1. La conversion des signaux à amplitude continue en signaux à amplitude discrète, ce processus met en jeu l'introduction d'un bruit généralement appelé "bruit de conversion A/N.
2. La conversion des coefficients de filtres obtenus par les procédures de réalisation de filtres discutées au chapitre 1, en coefficients de longueur de mot finie; il s'y associe une modification de la réponse en fréquence.
3. L'exécution des opérations (comme la multiplication et l'addition) de telle manière que la longueur de mot n'augmente pas de façon indésirable. Cela introduit un certain bruit et peut même causer des oscillations.

Dans toutes ces situations, nous avons affaire à la quantification ou au dépassement, voire aux deux phénomènes. Nous allons étudier ces effets plus en détails à la section 2.3.

2.2 Représentation des nombres dans un calculateur numérique **[16][17]**

Pour entamer cette section, on fait un petit rappel sur les trois représentations binaires les plus courantes :

1. En signe et amplitude (ou valeur absolue)
2. En complément à un
3. En complément à deux.

Les différences entre ces trois cas peuvent être illustrées très facilement à l'aide du tableau 2.1, qui montre la nature de la relation entre les huit mots de 3 bits possibles (B=3) et les valeurs décimales correspondantes.

Tableau 2.1 : Différentes représentations de nombres binaires

Valeur décimale	Signe et amplitude	Complément à un	Complément à deux
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
-0	100	111	-----
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	-----	----	100

Les valeurs décimales positives sont représentées de la même manière dans les trois représentations:

- Le bit le plus loin à gauche (le "bit de signe") est 0 si on a affaire à un nombre positif.
- Le bit le plus loin à droite (le bit "bit de plus faible poids" ou "bit le moins significatif" ou BMS) représente la valeur 2^0=1.
- Le second bit en partant de la droite représente la valeur 2^1=2.
- Si la longueur de mot est supérieure à 3, le troisième bit en partant de la droite représente la valeur 2^2 = 4, et ainsi de suite.

Ainsi: $01101 = + (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) = +13$.

Les nombres décimaux négatifs sont représentés d'une façon différente dans chacune des représentations, mais le bit de signe est toujours 1. Dans la représentation en signe et en amplitude, les autres bits représentent l'amplitude du nombre. Dans la représentation en complément à un, les nombres négatifs sont obtenus en remplaçant chacun des bits du nombre positif correspondant par le bit opposé ("inversion de bit"). Les nombres négatifs, dans la représentation en complément à deux, sont obtenus en inversant tous les bits du nombre positif correspondant et en ajoutant un "1" à la place correspondante au bit de plus faible poids.

Dans le tableau 2.1, on a assigné au bit de plus faible poids la valeur 2^0, cela signifie que nous pouvons seulement représenter des entiers. Cependant, nous pouvons aussi attribuer à ce bit de plus faible poids une valeur correspondant à une puissance

entière négative de 2, par exemple 2^{-3}=1/8. Nous pouvons alors représenter des fractions décimales par des mots binaires. On peut illustrer ceci en donnant la représentation des nombres décimaux +3.625 et -3.625 dans chacune des trois représentations avec une longueur de mot de 8 bits (Tableau 2.2).

Valeur décimale	**Signe et amplitude**	**Complément à un**	**Complément à deux**
+3.625	00011.101	00011.101	00011.101

Parmi les trois représentations données jusqu'à présent, la première et la troisième sont les plus utilisées. La représentation en signe et amplitude a l'avantage de réaliser des multiplications, la représentation en complément à deux a ses avantages pour les additions et les soustractions.

Une caractéristique agréable de la représentation en complément à deux est que les résultats intermédiaires dans une longue série d'additions p uans ymte M œ lep ne

La partie fractionnaire, pour un nombre représenté sur N bits prend la valeur suivante :

$$m = N - n \tag{2.4}$$

b- Résolution ou quantification du codage en virgule fixe **[6]**

La résolution ou quantification du codage, représente la variation la plus petite entre deux nombres successifs :

$$q = 2^{-m} \tag{2.5}$$

c- Domaine de définition d'un nombre en virgule fixe au formant [n.m]

Le domaine de définition des nombres définis selon le format [n.m] correspond aux valeurs extrêmes représentables.

$$D = [-2^{n-1} \dots 2^{n-1} - 2^{-m}] \tag{2.6}$$

d- Erreur due à la quantification **[18]**

Le nombre de bits pour exprimer un nombre fractionnaire étant limité, la reconstitution du nombre à partir de son expression en binaire ne donne pas une valeur identique au nombre d'origine. Il y a donc une erreur due à la quantification.

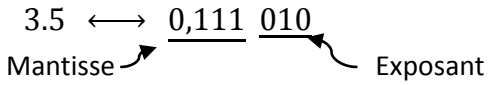
2.2.2 Représentation en virgule flottante

Une manière totalement différente d'attribuer une valeur décimale à un mot binaire est utilisée dans les représentations à virgule flottante. Ici, un nombre décimal A est d'abord écrit sous la forme :

$$A = M \times 2^E \tag{2.7}$$

Où E est un entier positif ou négatif et $0.5 \leq |M| < 1$ M est appelé la mantisse et E l'exposant. Chacun est représenté comme un mot binaire dans l'une des notations à virgule fixe, la mantisse ayant toujours un seul bit (le bit de signe) avant la virgule. Les deux mots pris ensemble représentent le nombre décimal. Comme exemple : Le nombre 3.5 est

d'abord réécrit (+0.875) x $2^{(+2)}$, avec une mantisse de 4 bits et un exposant de 3 bits, cela donne :



Le grand avantage de ce type de représentation est qu'il est possible de représenter une large gamme de nombre ; les petits nombres sont rapprochés les uns des autres (par exemple 4/64, 5/64, 6/64 et 7/64) alors que des nombres sont bien séparés (par exemple, 4, 5, 6 et 7). Cela signifie que des nombres successifs ont à peu prés le même espacement relatif tout au long de la gamme.

Un désavantage de la représentation à virgule flottante est que les opérations sont plus compliquées. Pour multiplier deux nombres, on doit multiplier les deux mantisses et additionner les deux exposants. Il faut alors s'assurer que le résultat remplit la condition (2.1), en ajustant si nécessaire la mantisse et l'exposant obtenus. Lorsqu'on additionne deux nombres, on doit d'abord s'assurer que les exposants des deux nombres sont les mêmes. Il faut ensuite additionner les mantisses et, si nécessaire, ajuster le résultat de nouveau pour que la condition (2.1) soit vérifiée. Ces opérations sont nettement plus compliquées que celles qu'on doit traiter dans les représentations à virgule fixe.

Dans la pratique, la seule fois où nous rencontrerons les représentations à virgule flottante sera lors de l'utilisation d'un ordinateur pour le traitement numérique des signaux, dans les autres cas (par exemple pour les filtres numériques sous forme de circuits intégrés spécialement conçus) on utilise généralement les représentations à virgule fixe.

L'IEEE (Institute of Electrical and Electronics Engineers) a développé un standard (IEEE 754) pour le calcul arithmétique en virgule flottante. Ce standard spécifie la manière de représenter des nombres en simple précision (32 bits) et double précision (64 bits).

a- Représentation simple précision **[19][20]**

Le standard IEEE 754 pour les nombres en virgule flottante en simple précision est de 32 bits. Le premier bit S (le plus à gauche) représente le signe, les 8 suivants (E)

représentent l'exposant, et les 23 derniers (F) représentent la partie fractionnaire du nombre, appelé mantisse.

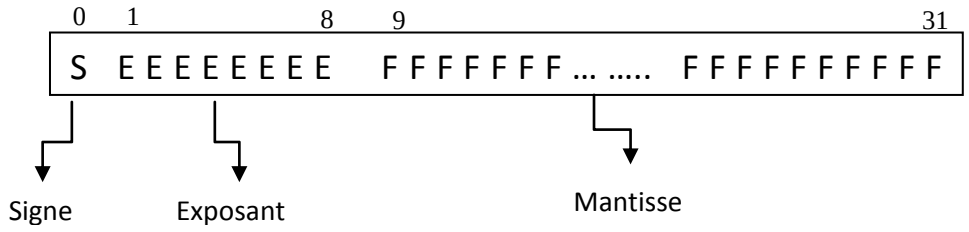


Figure 2.2.a Représentation d'un nombre binaire en virgule flottante, simple précision

b- Représentation double précision **[19][20]**

La virgule flottante double précision, selon le standard IEEE est constituée d'un mot de 64 bits. Le premier bit S (le plus à gauche) représente le signe, les 11 bits suivants (E) représentent l'exposant, et les 52 derniers (F) représentent la partie fractionnaire du nombre, appelée mantisse.

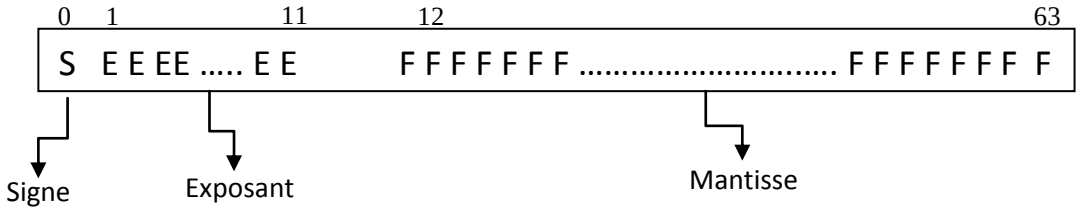


Figure 2.2.b Représentation d'un nombre binaire en virgule flottante, double précision

c- Résolution ou quantification du codage en virgule flottante

La résolution ou quantification du codage en virgule flottante dépend de la valeur de l'exposant E. Pour un exposant de n_E bits et une mantisse de n_m bits, soit un codage en virgule flottante de ($N = n_m + 2 + n_E$) bits et pour un exposant $0 < E < 2^{nE}$:

$$q = 2^{2^{nE}-1}.2^{-m_E} \tag{2.8}$$

d- Domaine de définition d'un nombre en virgule flottante

Le domaine de définition des nombres en virgule flottante avec un exposant de n_E bits et une mantisse de n_m bits, pour un exposant $0 < E < 2^{n_E-1}$ est donné par :

$$D =]-2^{(n_E-1)} \ldots . 2^{(n_E-1)}[\quad (2.9)$$

2.2.1 Comparaison virgule flottante et virgule fixe **[6]**

De nombreuses architectures numériques sont utilisées pour mettre en œuvre les systèmes d'asservissement, de régulation ou de traitement du signal (microcontrôleurs, microprocesseurs, processeurs de traitement du signal, etc...). Mais les calculs qui y sont réalisés ne le sont que de deux manières : les calculs en virgule fixe (lorsque les grandeurs sont représentées en virgule fixe) et les calculs en virgule flottante.

Dans cette section, une comparaison entre les codages en virgule flottante et virgule fixe est présentée. Cette analyse est faite selon deux points de vue, l'aspect dynamique et l'aspect implémentation.

a) Comparaison de la dynamique:

La dynamique D_{dB} d'un code correspond au rapport logarithmique entre les valeurs maximales x_{max} et minimales x_{min} de ce code **[6]**:

$$D_{dB} = 20 \log\left(\frac{x_{max}}{x_{min}}\right) \quad (2.10)$$

La dynamique permet d'illustrer l'espace des valeurs représentables par le code. L'intérêt d'une dynamique importante est d'assurer l'absence de débordement.
Dans le cas de l'arithmétique virgule flottante, la dynamique des données s'exprime sous la forme suivante:
Pour un codage en virgule flottante de N bits, en format n_E bits pour l'exposant, et N-(n_E-1) pour la mantisse, on obtient pour $0 < E < 2^{nE}$, et selon la définition :

$$\left.\begin{matrix} max(|A|) = 2^{(2^{n_E}-1)} \\ min(|A|) = 2^{(-2^{n_E}-1)} \end{matrix}\right\} \ , D_N = 20\log_{10}\left(\frac{2^{(2^{n_E}-1)}}{2^{(-2^{n_E}-1)}}\right) = 20(2^{2^{n_E}-1})\log_{10}(2)$$

$$D_{dB} \cong 20\log(2^{2K+1}+1) \qquad \text{Avec} \qquad K = 2^{n_E-1}-1 \qquad (2.11)$$

Où n_E désigne le nombre de bits alloués pour l'exposant.

Dans le cas de l'arithmétique virgule fixe, la dynamique est donnée par la relation suivante :

Pour un codage en virgule fixe de N bits en format n.m (m bits pour la partie entière et le signe, et n bits pour la partie fractionnaire, on peut écrire :

$$\left.\begin{matrix} \max(|A|) = 2^{n-1} \\ \min(|A|) = 2^{-m} \end{matrix}\right\}, D_N = 20\log_{10}\left(\frac{2^{n-1}}{2^{-m}}\right) = 20\log_{10}\left(2^{n+m-1}\right) = 20(N-1)\log_{10}(2)$$

$$D_{dB} = 20(b-1)\log(2) = 6(b-1) \qquad (2.12)$$

L'évolution de la dynamique pour chacune des deux arithmétiques en fonction de l'évolution du nombre de bits alloués est représentée à la figure 2.3 **[6]**. La dynamique de l'arithmétique virgule fixe est bien linéaire comme défini à l'équation (2.12). De plus, pour un nombre de bits inférieur à 16, la dynamique de l'arithmétique virgule fixe est supérieure à la dynamique de l'arithmétique virgule flottante. Par contre, pour un nombre de bits supérieur à 16, l'arithmétique virgule flottante a une dynamique supérieure. Pour un nombre de bits élevé (supérieur à 25), l'arithmétique virgule flottante montre son intérêt par sa dynamique très importante.

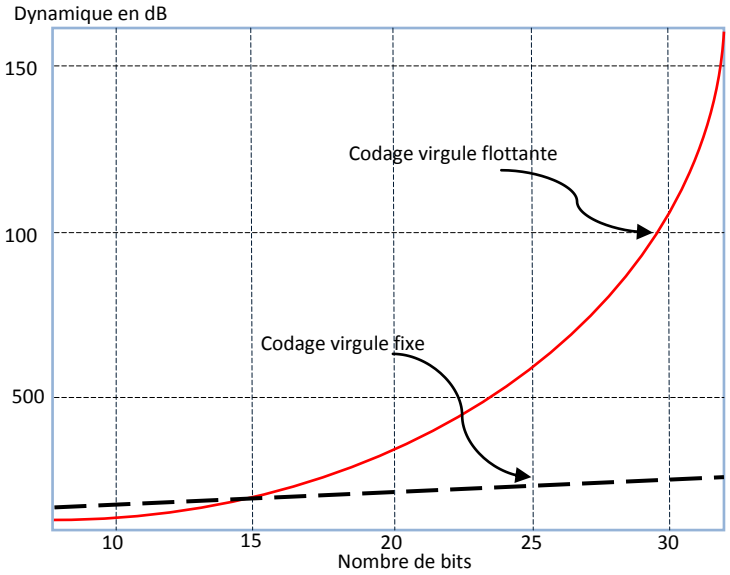


Figure 2.3 Evolution du niveau de la dynamique des codages en virgule fixe et en virgule flottante en fonction du nombre de bits utilisés.

b) Comparaison au niveau implantation

Le choix de l'arithmétique pour un système est fonction des contraintes associées à l'implantation. L'arithmétique virgule flottante présente l'avantage d'avoir une plus grande dynamique pour des données de taille 32 bits ainsi qu'un meilleur rapport signal/bruit (RSB) par rapport à l'arithmétique virgule fixe. Néanmoins, comme l'exposant est variable pour les différents données codées, les opérations en arithmétique virgule flottante sont complexes à effectuer. Par exemple, l'addition en virgule flottante s'effectue en trois étapes. Tout d'abord, les données sont non-normalisées pour obtenir leur valeur réelle. Ensuite, l'addition est effectuée. Enfin, la normalisation est mise en œuvre pour adapter l'exposant à la valeur de la sortie de l'addition. Ainsi, le coût d'une opération en virgule flottante est très élevé par rapport au coût d'une opération en virgule fixe pour laquelle les bits sont simplement décalés pour aligner la virgule dans le cas d'une addition. Les systèmes embarqués doivent avoir une consommation en énergie très limitée obtenue grâce à la virgule fixe. En effet, l'arithmétique virgule flottante basée sur le norme IEEE-754 **[16]** nécessite d'utiliser des données codées sur au moins 32 bits. Or, la majorité des données virgule fixe sont codées sur 16 bits. Ainsi, la largeur des bus et des mémoires au sein des architectures virgule fixe sont plus faibles. Par conséquent, ajouté au fait que les opérateurs en virgule fixe sont moins complexes, la consommation d'énergie et le coût sont moins importants pour une architecture basée sur l'arithmétique virgule fixe.

De ce fait, dans le cadre d'une implantation de l'algorithme dans un système embarqué, l'arithmétique virgule fixe sera privilégiée.
Cependant, l'arithmétique virgule fixe présente le désavantage d'avoir un RSB plus faible et donc une précision des calculs moins bonne. Ainsi, il est nécessaire de porter attention à la précision des calculs lors d'une implantation en virgule fixe.

2.3 Analyse des effets de la précision finie **[15]**

Les étapes d'approximation et de réalisation supposent qu'on travaille en précision infinie ou très élevée. Cependant dans l'étape d'implémentation, il est souvent nécessaire de représenter les coefficients du filtre en utilisant un nombre limité de bits, typiquement 16bits à 32bits et les opérations arithmétiques indiquées dans l'équation aux différences sont réalisés en utilisant une arithmétique à précision finie.

Les effets de l'utilisation d'un nombre fini de bits, dégradent les performances du filtre, et dans certains cas, le rendent instable.

Le concepteur doit analyser ces effets et choisir la longueur de mots adéquate (nombre de bits) pour les coefficients du filtre, les variables du filtre qui sont les échantillons des signaux d'entrée et de sortie et enfin pour les opérations arithmétiques à l'intérieur du filtre.
Comme c'est indiqué dans l'introduction de ce chapitre, les principales sources de dégradation de performances dans les filtres numériques sont :

1- Quantification des signaux d'entrée et de sortie ;
2- Quantification des coefficients du filtre ;
3- Erreurs de l'arithmétique utilisée (troncature et arrondi) ;
4- Dépassement (overflow).

2.3.1 Quantification des signaux d'entrée et sortie

Dans un système numérique, la première occasion où on rencontre une quantification est habituellement liée à la conversion d'un signal d'entrée analogique en un signal d'entrée numérique. A coté du passage du temps continu au temps discret (par échantillonnage), on doit aussi transformer une amplitude continue en une amplitude discrète (par quantification). On effectue souvent dans ce cas un arrondi (figure 2.4).

Chaque échantillon d'entrée *x(n)* est converti en un échantillon de sortie $x_q(n)$. Ce faisant, on introduit une erreur *e(n)*, donnée par:

$$e(n) = x_q(n) - x(n)$$

2.3.2 Quantification des coefficients du filtre **[15]**

A l'aide des procédures de réalisation de filtres numériques que nous avons décrites au chapitre 1, on trouve généralement les valeurs des coefficients des filtres avec une très grande précision. Pour obtenir un filtre numérique réalisable dans lequel les coefficients n'ont qu'une longueur de mot finie, il faut quantifier ces valeurs. Mais ce faisant, on modifie la réponse en fréquence du filtre et même, la position des pôles et des zéros du filtre. Ces modifications peuvent être assez importantes. Il peut arriver qu'après quantification, le filtre ne satisfait plus les spécifications sur lesquelles le calcul des coefficients non-quantifiés était basé.

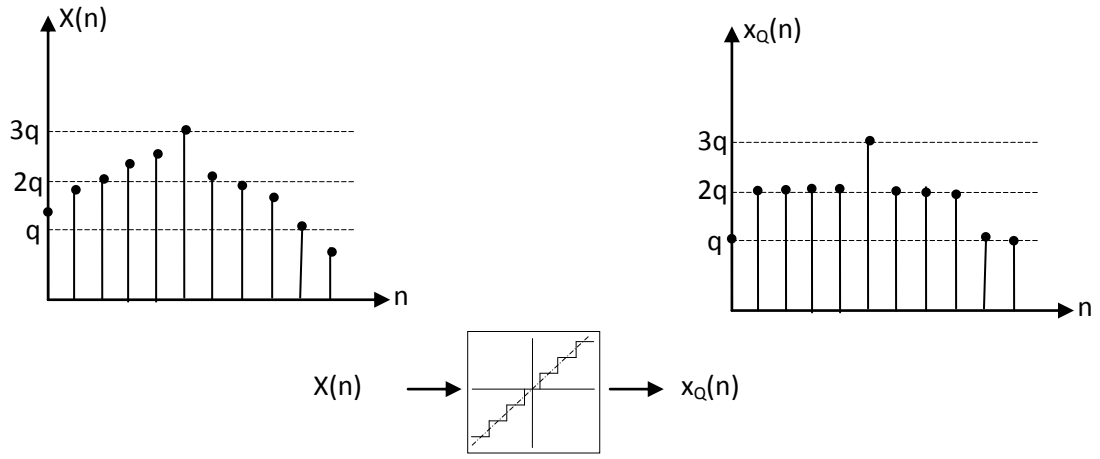


Figure 2.4 : Quantification en conversion analogique-numérique

Dans des cas extrêmes, un filtre stable peut même devenir instable, si un pôle se décale d'une position à l'intérieur du cercle unité du plan des z vers une position à l'extérieur. La quantification des coefficients n'introduit pas de non linéarités ni d'effets qui dépendent du signal d'entrée ou varient avec le temps. Elle a seulement pour résultat une modification "une fois pour toutes" des propriétés du filtre.

On peut analyser cette modification dans les détails en calculant la réponse impulsionnelle ou la réponse en fréquence après quantification. On peut alors savoir si le filtre rencontre encore la spécification de réalisation. C'est une bonne habitude que de rendre les spécifications plus serrées qu'il n'est strictement nécessaire pour le calcul de départ sans quantification, de sorte qu'il y ait une marge pour les erreurs qui surviennent en raison de la quantification.

On a trouvé que certaines structures de filtres sont plus sensibles que d'autres à la quantification. En général, on peut dire qu'une structure est d'autant moins sensible que la position de chaque pôle et de chaque zéro dépend de moins de coefficients:

1. Dans la forme directe 1 et la forme directe 2 chaque pôle est affecté par les valeurs de tous les coefficients a_i et de même chaque zéro par les valeurs de tous les coefficients b_i par conséquent, de petites modifications dans les a_i et les b_i peuvent produire de grands décalages de pôles et des zéros. C'est en particulier le cas si les pôles ou les zéros ou encore les deux sont rapprochés les uns des autres, comme dans les filtres à bande étroite par exemple.

2. Dans la structure parallèle, chacun des pôles (ou chacune des paires de pôles) est déterminé par un petit nombre de coefficients dans chaque branche parallèle. Alors que les zéros surviennent par suite d'annulation entre les signaux de sortie des différentes branches et dépendent donc de tous les coefficients. Ce type de filtre est donc assez insensible dans la bande passante (qui est surtout déterminée par les pôles), mais très sensible dans la bande affaiblie (surtout déterminée par les zéros).

3. Dans la structure en cascade, aussi bien les pôles que les zéros sont déterminés par un petit nombre de coefficients. Les filtres qui possèdent cette structure sont donc peu sensibles aussi bien dans la bande passante que dans la bande affaiblie. On trouve aussi une propriété importante dans la connexion en cascade de cellules du premier et du second ordre ayant toutes deux une structure de forme directe. Les zéros sur le cercle unité y restent après quantification, bien qu'on puisse parfois noter un léger décalage.

2.3.3 Erreurs de l'arithmétique utilisée **[15][20]**

Le concept de "*quantification*" est le processus par lequel une quantité x est convertie en une quantité x_q qui est approximativement égale à x mais qui peut prendre moins de valeurs différentes que x. Ainsi, substituer à tout nombre réel l'entier le plus proche relève de la quantification (4 au lieu de 3.67 par exemple). On peut citer comme autre exemple la réduction de la longueur du mot d'une quantité binaire x en diminuant le nombre de bits après la virgule (1011,1101111 remplacé par 1011,110). La relation entre x et x_q est appelée caractéristique de quantification. Les formes de quantifications les plus largement répandues en traitement numérique du signal sont:

1. L'arrondi.
2. La troncature en valeur.
3. La troncature en amplitude.

La figure 2.5 représente les caractéristiques de quantification correspondant à ces trois cas. Les valeurs successives possibles pour x_q sont séparées par une distance fixe q (échelon de quantification). On voit que la conversion de x en x_q est presque toujours associée à l'introduction d'une inexactitude. L'amplitude de cette inexactitude, $x_q - x$, est cependant limitée et on peut l'exprimer en fonction de q:

1. Pour l'arrondi:

$$-q/2 \leq x_q - x \leq q/2 \tag{2.13}$$

2. Pour la troncature en valeur:

$$-q \leq x_q - x < 0 \tag{2.14}$$

3. Pour la troncature en amplitude:

$$\begin{aligned} -q \leq x_q - x < 0 \qquad & si \quad x > 0 \\ 0 \leq x_q - x < q \qquad & si \quad x < 0 \end{aligned} \tag{2.15}$$

La quantification est par essence une opération non linéaire puisqu'en général:

$$(x+y)_q \neq x_q + y_q$$

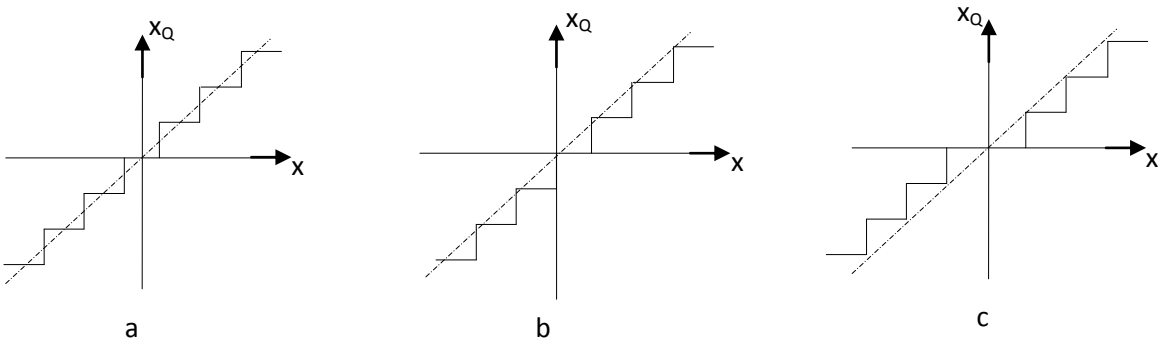


Figure 2.5 : Les trois caractéristiques de quantification: (a) Arrondi, (b) Troncature en valeur, (c) Troncature en amplitude

Une autre forme de non linéarité qu'il peut nous arriver de rencontrer dans les signaux et les systèmes numériques est connue sous le nom de dépassement. Le dépassement est ce qui se produit lorsqu'une quantité *x* cherche à atteindre une valeur en dehors des limites (-X et X) qu'on doit observer. En termes formels, on peut décrire ce phénomène comme la conversion de *x* en x_p, où:

$$x_p = x \qquad si \qquad |x| \leq X$$
$$-X \leq x_p \leq X \qquad si \qquad |x| > X$$

La relation entre *x* et x_p est appelée caractéristique de dépassement. Nous en donnons trois exemples en figure 2.6:

1. La saturation
2. La mise à zéro
3. Le dépassement en "dents de scie".

En traitement numérique du signal, on peut rencontrer un dépassement si on veut réduire la longueur de mot d'une quantité binaire *x* en diminuant le nombre de bits avant la virgule (par exemple à 1011,110111 on substitue 11,110111).

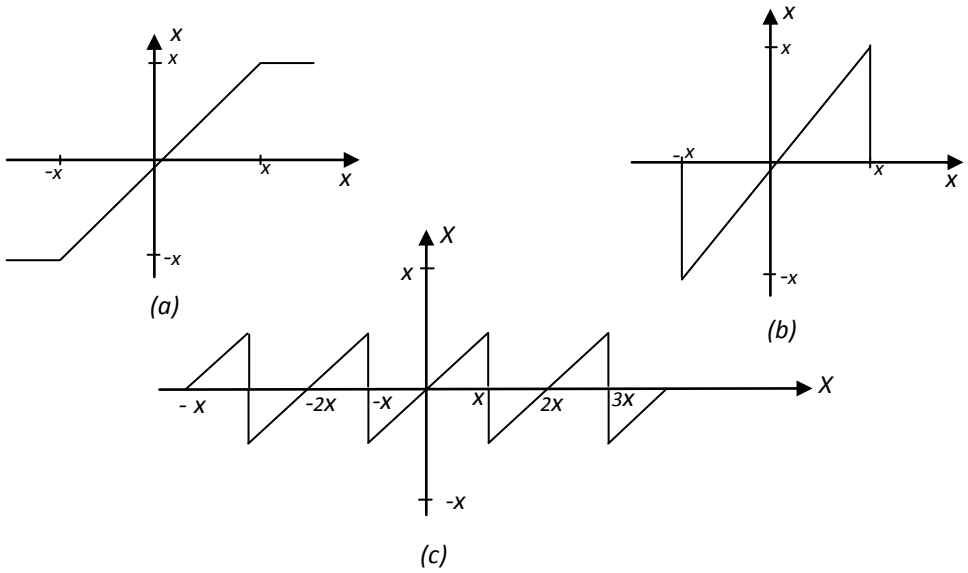


Figure 2.6 : Les trois caractéristiques de dépassement ; (a) Saturation, (b) Mise à zéro, (c) Dents de scie

Dans certains cas, il ya combinaison de quantification et dépassement. Cela se produit par exemple si on veut substituer à tout nombre réel l'entier le plus proche compris entre +99 et -99. Pour les nombres dont la valeur absolue est inferieur à 99, on a une quantification (arrondi) et pour les autres valeurs, il s'agit de dépassement (saturation). Ces deux effets peuvent être représentés sur une même caractéristique x_F.

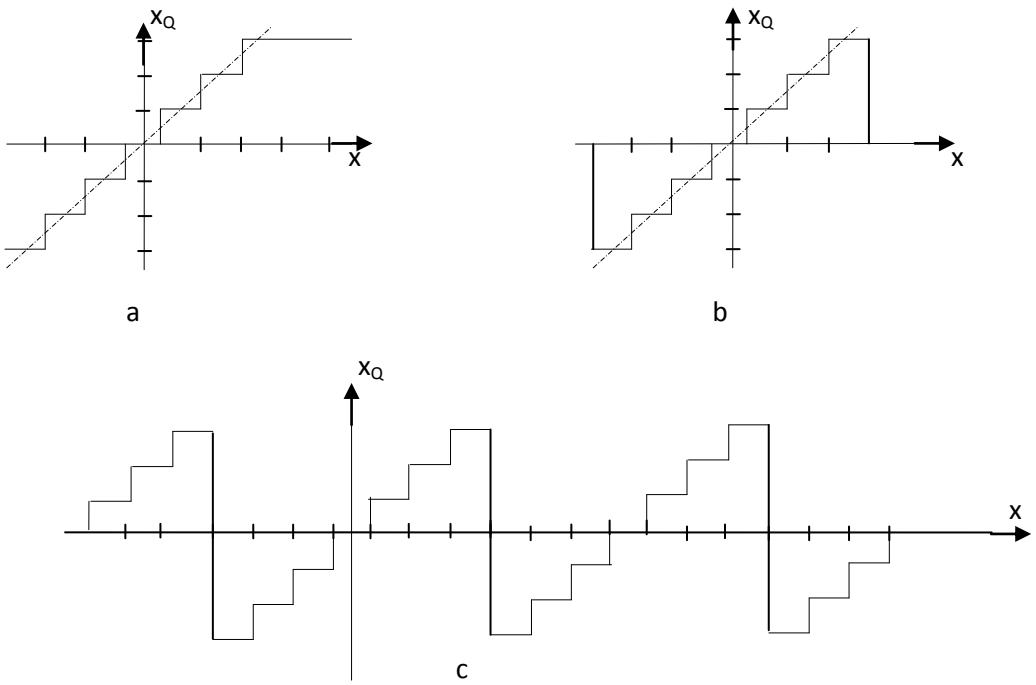


Figure 2.7 : Combinaison de quantification (arrondi) et de différents types de dépassement.

2.4 Limitation de la longueur de mot des résultats intermédiaires **[21]**

C'est dans la limitation de la longueur de mot des résultats intermédiaires dans les systèmes numériques qu'on trouve les conséquences les plus compliquées du fait de travailler sur des mots de longueur finie. Par résultat intermédiaire, nous désignons le résultat d'une addition ou d'une multiplication effectuée quelque part dans le système numérique. Nous supposons ici que nous voulons réaliser un système numérique dans lequel toutes les quantités sont représentées par des mots de B bits dans une représentation à virgule fixe, avec seulement un bit (le bit de signe) avant la virgule. Nous ne pouvons donc représenter que des quantités dont la valeur absolue est inferieure à 1.

2.4.1 Dépassement des résultats intermédiaires **[21]**

Le dépassement dans un filtre numérique peut induire des erreurs considérables dans le signal de sortie, par comparaison avec le signal de sortie du filtre linéaire qu'on aurait aimé avoir en réalité. Dans un filtre de structure non-récursive, ces erreurs ont une durée finie (jamais plus longue que la durée de la réponse impulsionnelle). Dans un filtre récursif, si un dépassement se produit une seule fois, cela peut durer indéfiniment et induire toutes sortes d'effets indésirables. Nous citons ci-dessous, un certain nombre d'effets :

1. Si le signal d'entrée $x(n)$ est nul à partir d'une certaine valeur de n après l'apparition du dépassement, il peut se produire une oscillation permanente (oscillation de dépassement). Cette oscillation a une grande amplitude reliée au niveau de dépassement.
2. Si le signal d'entrée $x(n)$ est périodique, on peut obtenir pour un même signal d'entrée des signaux de sortie complètement différents par suite du dépassement, et qui dépendent des états initiaux des registres du filtre.
3. Si le signal d'entrée est périodique, de petites modifications du signal d'entrée peuvent induire de grandes différences dans le signal de sortie par suite de dépassement (phénomène de saut).

Au total, il existe beaucoup de raisons d'éviter le dépassement. Pour ce faire, on effectue un recadrage ou encore une mise à l'échelle (scaling en anglais). Cela signifie que le signal d'entrée du filtre est multiplié par un facteur $S<1$ de sorte qu'il ne peut plus se

produire de dépassement. Il est préférable d'utiliser ici une puissance entière de 2 (par exemple $2^{-2} = 0.25$) parce que la multiplication se résume alors à un décalage des bits.

2.4.2 Quantification des résultats intermédiaires

Considérons maintenant des systèmes numériques dans lesquels il ne peut se produire de dépassement, grâce à une mise à échelle correcte. Chaque quantification d'un résultat intermédiaire correspond à l'introduction d'une inexactitude, dans les limites données par l'une des équations (2.13), (2.14) ou (2.15). Si les valeurs successives des résultats intermédiaires sont suffisamment aléatoires, on peut considérer ces inexactitudes comme un signal interférant (bruit de quantification) qui s'ajoute au signal utile à l'endroit où le quantificateur est localisé.

2.5 Conclusion

Après tout ce que l'on a dit au sujet de la limitation de la longueur de mot des résultats intermédiaires, il est clair que la réalisation d'un filtre numérique sera loin d'être achevée quand on aura trouvé les valeurs des coefficients non quantifiés, en utilisant les notions du chapitre 1. L'étape suivante sera alors la quantification des coefficients de telle sorte que les spécifications soient satisfaites. On analyse ensuite les effets que peut produire la limitation de la longueur de mot des résultats intermédiaires et on les minimise autant que possible. En connectant en cascade des cellules du second ordre, ou en regroupant les pôles et les zéros d'une façon adéquate **[5]**.

CHAPITRE 3

LES ALGORITHMES GENETIQUES, PRINCIPE ET APPLICATIONS

3.1 Introduction **[22]**

Depuis Euclide (450 ans avant J.C) les mathématiques n'ont pas cessé de modéliser les phénomènes naturels, en mettant au point des concepts de plus en plus raffinés pour prendre en compte les multiples aspects de ces derniers. D'après *Darwin ``La vie est une compétition et les mieux adaptes survivent et se reproduisent``*, cette règle qui a engendrée les organismes que nous connaissons aujourd'hui, est rendue utile pour la résolution des problèmes d'optimisation dans divers domaines de la science, grâce a l'orientation de la recherche vers des techniques d'optimisation basées sur les analogies avec des processus naturels, biologiques, ou humains. Parmi ces techniques, on trouve les *Algorithmes Evolutionnaires (AE)* qui sont des procédures stochastiques (pseudo-aléatoire) inspirées des lois de l'évolution des espèces, et de la génétique Naturelle. *Les Algorithmes Evolutionnaires* font partie du champ de *l'Intelligence Artificielle (IA).* Il s'agit d'*IA* de bas niveau, inspirée par " l'intelligence " de la Nature. En réalité trois types d'AE on été développés isolément et à peu prés simultanément, dans les années 60, par différents scientifiques : les *Algorithmes Génétiques*, les *Stratégies d'Evolution*. , *Programmation Evolutionnaire*. Présentant des différences marquées à l'origine, ils tendent de plus en plus à se confondre suite à leurs emprunts respectifs. Dans les années 90, ces trois champs ont commencé à sortir de leur isolement et ont été regroupés sous le terme anglo-saxon d'*Evolutionnary Computation*. Parmi les AE que nous venons de citer, nous avons choisi de traiter des *Algorithmes Génétiques (AG).* En effet, ils nous paraissaient concilier au mieux puissance, généralité et facilité de programmation. Leur particularité est qu'ils sont fondés sur le *Néo-Darwinisme*, c'est-à-dire l'union de la théorie de l'évolution et de la génétique moderne. Ainsi, les variables sont généralement codées en binaire ou en réel (par analogie avec les quatre lettres de l'alphabet génétique) sous forme de gènes dans un chromosome. Des opérateurs génétiques (croisement, mutation) sont appliqués à ces chaînes qui sont les chromosomes.

L’AG permet d’obtenir des solutions à un problème n’ayant pas des méthodes de résolution décrites précisément, ou dont la solution exacte, si elle est connue, est trop compliquée pour être calculée en un temps raisonnable.

Dans ce cadre, On peut citer quelques problèmes complexes comme : l'optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), le traitement d’image (alignement de images satellites, reconnaissance de suspects…), l'optimisation d’emplois du temps, l'optimisation de design, le contrôle de systèmes industriels et l'apprentissage des réseaux de neurones **[23]**.

De même, les AGs peuvent être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire…) car la population peut s’adapter à des conditions changeantes. En particulier, ils supportent bien l’existence de bruit dans la fonction à optimiser. Ils peuvent aussi servir à déterminer la configuration d’énergie minimale d’une molécule ou à modéliser le comportement animal.

3.2 Historique **[24]**

1860 - Charles Darwin publie son livre intitulé *L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature*. Dans ce livre, Darwin rejette l'existence «de systèmes naturels figés», déjà adaptés pour toujours à toutes les conditions extérieures, et expose sa théorie de l'évolution des espèces : sous l'influence des contraintes extérieurs, les êtres vivants se sont graduellement adaptés à leur milieu naturel au travers de processus de reproductions.

20ième siècle - Mise en évidence de l'existence de mutations génétiques. Les problèmes de traitement de l'information sont résolus de manières figés : lors de sa phase de conception, le système reçoit toutes les caractéristiques nécessaires pour les conditions d'exploitations connues au moment de sa conception ce qui empêche une adaptation à des conditions d'environnement inconnues, variables ou évolutives. Les chercheurs en informatique étudient donc des méthodes pour permettre aux systèmes d'évoluer spontanément en fonction de nouvelles conditions.

1966 - Programmation évolutionnaire L. J. Fogel.

1973 - Stratégie d'évolution I. Rechenberg.

1975 - Dans les années 1960, John Holland étudient les systèmes évolutifs et, en 1975, il introduit le premier modèle formel des algorithmes génétiques (*the canonical genetic algorithm AGC*) dans son livre *Adaptation in Natural and Artificial Systems*. Ce modèle servira de base aux recherches ultérieures.

1989 - David Goldberg publie un ouvrage de vulgarisation des algorithmes génétiques.

Années 90 - Programmation d'une panoplie d'algorithmes génétiques transcris en C++, appelée GAlib. Cette librairie contient des outils pour des problèmes d'optimisation en utilisant les AG. Elle est conçue pour servir de support de programmation.

3.3 Lexique des termes

On définie les termes importants du domaine de l'algorithme génétique en établissant une analogie avec le monde de la biologie.

Tableau 3-1 : les termes de base de l'algorithme génétique.

Domaine / **Paramètres**	Biologie	Algorithme génétique
Chromosome	Support physique des informations héréditaires (caractéristiques) d'un organisme vivant.	Une Chaîne codée représentant une solution potentielle appelée également ``individu``.
Gène	Représente une caractéristique (ex: le gène de la couleur des yeux).	Un bit ou ensemble de bits codant une information.
Allèle	La valeur d'une caractéristique (ex: Bleu pour la couleur des yeux).	Valeur de la suite des bits (ex:«010» représente un entier sa valeur égale à 2).
Locus	La position du gène dans le chromosome.	Position de bloc de bits dans la chaîne binaire.
Génotype	La position du gène dans le chromosome.	Structure qui code la chaîne (ensemble de paramètres).
Phénotype	C'est l'organisme formé par l'interaction de l'ensemble du matériel génétique.	Représente la structure décodée, ou un point dans l'espace de solutions.

3.4 Principe de fonctionnement des AGs standard (AGS)

Un AG standard nécessite en premier, le codage de l'ensemble des paramètres du problème d'optimisation en une chaîne de longueur finie. Le principe d'un AG est simple, il s'agit de simuler l'évolution d'une population d'individus jusqu'à un critère d'arrêt. On commence par générer une population initiale d'individus (solutions) de façon aléatoire. Puis, à chaque génération, des individus sont sélectionnés, cette sélection est effectuée à partir d'une fonction objective appelée fonction d'adaptation. Puis, les opérateurs de croisement et de mutation sont appliqués, et une nouvelle population est créée. Ce processus est itéré jusqu'à un critère d'arrêt **[25]**. Le critère le plus couramment utilisé est le nombre maximal de générations que l'on désire effectuer. La figure 3.1 présente le principe de l'AG Standard. L'AG débute par la génération d'une population initiale et l'évaluation de la fonction d'adaptation de tous les individus qui composent cette première population. Puis, des individus sont sélectionnés aléatoirement pour la reproduction selon le principe de la survie du plus adapté. Ensuite, des individus « enfants » (ou descendants) sont générés, puis placés dans une population, cette dernière remplacera entièrement la population parents.

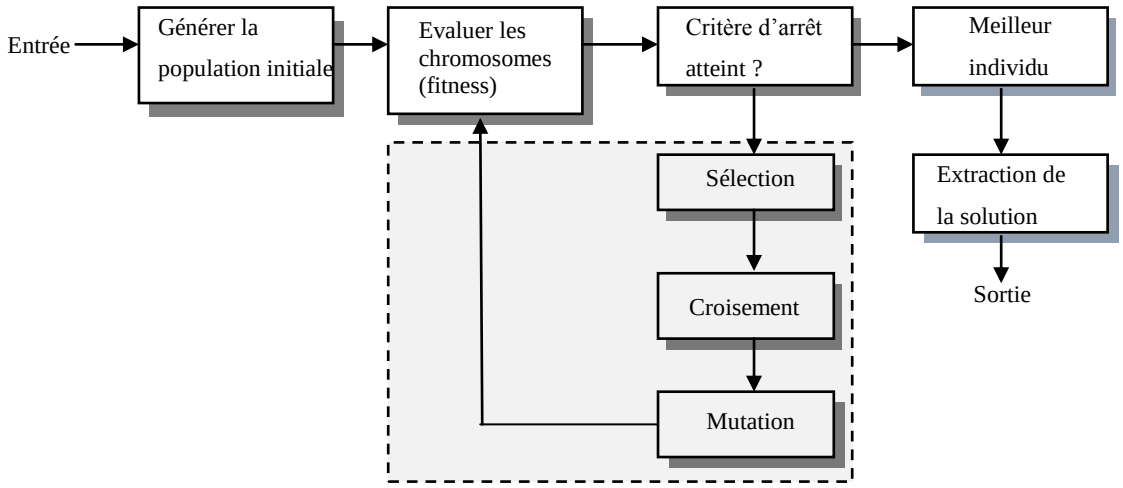


Figure 3.1 : Procédure classique d'un algorithme génétique

3.5 Les opérateurs génétiques **[23][25]**

Les opérateurs génétiques jouent un rôle prépondérant dans la possible réussite d'un AG. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement

et de mutation. Si le principe de chacun de ces opérateurs est facilement compréhensible, il est toutefois difficile d'expliquer l'importance isolée de chacun de ces opérateurs dans la réussite d'un AG. Cela tient pour partie au fait que chacun de ces opérateurs agit selon divers critères qui lui sont propres (valeur sélective des individus, probabilité d'activation de l'opérateur, etc…).

3.5.1 Sélection

Comme son nom l'indique, la sélection vise à élire une population parent (producteurs) à partir d'une population initiale. Les meilleurs individus peuvent être choisis plusieurs fois dans la prochaine génération, alors que les moins inaptes auront moins de chance de l'être. La sélection se fera aléatoirement en fonction de la qualité d'adéquation liée au problème posé. Ce choix est réalisé par tirage au sort parmi les N individus, en tenant compte d'une probabilité de sélection affectée à chacun d'eux.

Un individu a d'autant plus de chance d'être sélectionné que sa fonction d'évaluation $f_{éval}$ (terme anglo saxon : fitness) prend une valeur importante. Pratiquement, la probabilité p_i d'évolution d'un individu c_i est définie par :

$$p_i = \frac{f_{éval}(c_i)}{\sum_{j=1}^{N} f_{éval}(c_j)} \qquad (3.1)$$

Où $\sum_{j=1}^{N} f_{éval}(c_j)$: représente la somme de toutes les valeurs des fonctions d'évaluations de chaque chromosome $\mathbf{c}_i$ de la population.

Elle joue un rôle important dans l'accélération de la convergence de l'algorithme. Il existe plusieurs types de sélection, parmi eux, on peut citer :

a) La sélection par roulette (Wheel sélection).

Cette méthode a été initiée par J. Holland lui même en 1975. Elle procède en une sélection proportionnelle au niveau de la fonction d'évaluation $f_{éval}$ qui consiste à associer à chaque individu, une probabilité de sélection p_i définie par l'équation (3.1). On tire alors un nombre aléatoire, puis on regarde quel est le chromosome sélectionné. Ainsi les meilleurs individus seront plus adressés que les plus mauvais. Lors du tirage au sort, certains individus peuvent être retenus plusieurs fois alors que d'autres sont

tenus à l'écart. Ceci s'effectue par le calcul d'une probabilité de sélection cumulée q_i telle que :

$$q_i = \sum_{j=1}^{i} p_j = p_1 + p_2 + \cdots p_i \qquad (3.2)$$

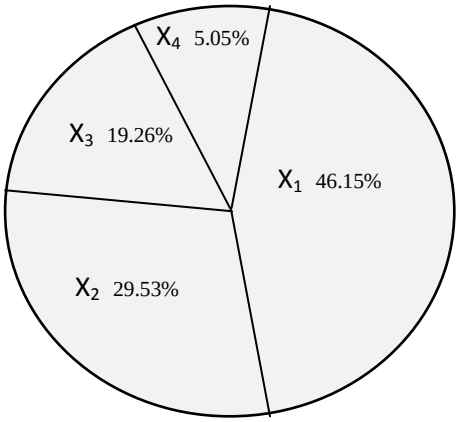


Figure 3.2 sélection par roulette pour une population de 4 individus

On génère aléatoirement (loi normale) un nombre réel r sur l'intervalle [0 , 1]. Cette valeur est générée plusieurs fois en fonction de la taille de la population. L'individu C_i est sélectionné lorsque $q_{i-1} < r < q_i$. Mais si la qualité d'adéquation d'un individu est très supérieure à la moyenne des $f_{éval}$, il constituera la majorité des individus de la population suivante, ce qui peut conduire à une perte de la diversification dans la population qui elle même , peut conduire à une convergence prématurée ou une diminution rapide de l'espace de recherche.

b) <u>La sélection élitiste.</u>

Dans cette technique, seule la moitié supérieure de la population N comprenant les meilleurs individus sont sélectionnés pour participer à la naissance de la génération suivante.

Néanmoins, on remarque qu'il y a une forte pression de sélection qui peut conduire à une convergence prématurée vers une solution non désirée et comme tout individu peut transmettre à sa descendance des gènes qui, une fois combinés avec d'autres peuvent se révéler intéressants, ceci peut induire à une diminution du caractère de diversification dans la population ou une dérive génétique.

c) La sélection par tournoi

Deux individus sont choisis au hasard et combattent (en comparant leurs fonctions d'adaptation) pour accéder à la génération intermédiaire. Le plus adapté l'emporte avec une probabilité, que nous avons généralement prise égale à 1 (une valeur inférieure permet de réduire la pression de sélection si nécessaire). Cette étape est répétée jusqu'à ce que la génération intermédiaire soit remplie (N/2 composants). Il est tout à fait possible que certains individus participent à plusieurs tournois : s'ils gagnent plusieurs fois, ils auront donc droit d'être copiés plusieurs fois dans la génération intermédiaire, ce qui favorisera la pérennité de leurs gènes.

3.5.2 Croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes (slicing crossover). Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents (A1 et A2). On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C_1 et C_2 (voir figure 3.3.a).

L'opération de croisement permet de créer de nouvelles chaînes en échangeant de l'information entre deux chaînes (figure 3.3.a). Le croisement s'effectue en deux étapes, d'abord les nouveaux éléments produits par la reproduction sont appariés, ensuite chaque paire de chaînes subit un croisement comme suit : un entier k représentant une position sur la chaîne est choisi aléatoirement entre 1 et (L-1) (L étant la longueur de la chaîne). Deux nouvelles chaînes sont créées en échangeant tous les caractères compris entre les positions k+1 et L inclusivement. L'exemple suivant (figure 3.3.a) montre deux chaînes (A1 et A2) de longueur L = 5 appartenant à la population initiale. Les deux nouvelles chaînes (C1 et C2) appartenant à la nouvelle population sont obtenues par croisement à la position k = 4.

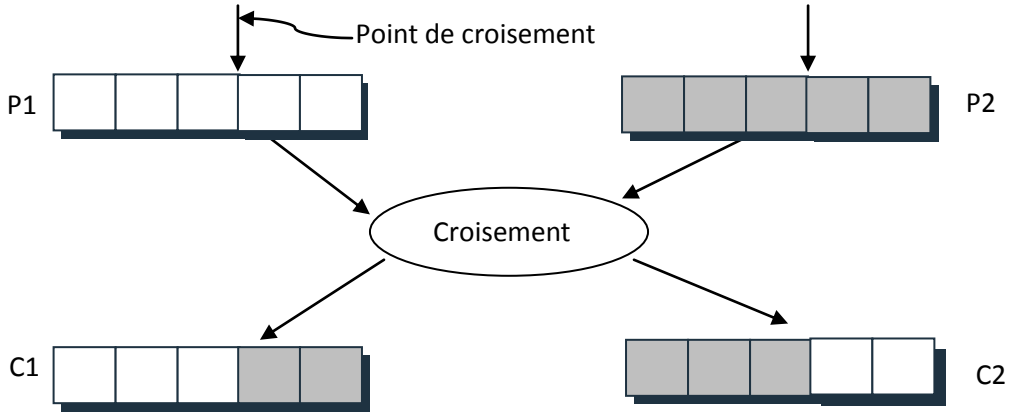


Figure 3.3.a Croisement à un point

On peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, ou 4 sous- chaînes ou plus **[26]**, (voir figure 3.3.b).

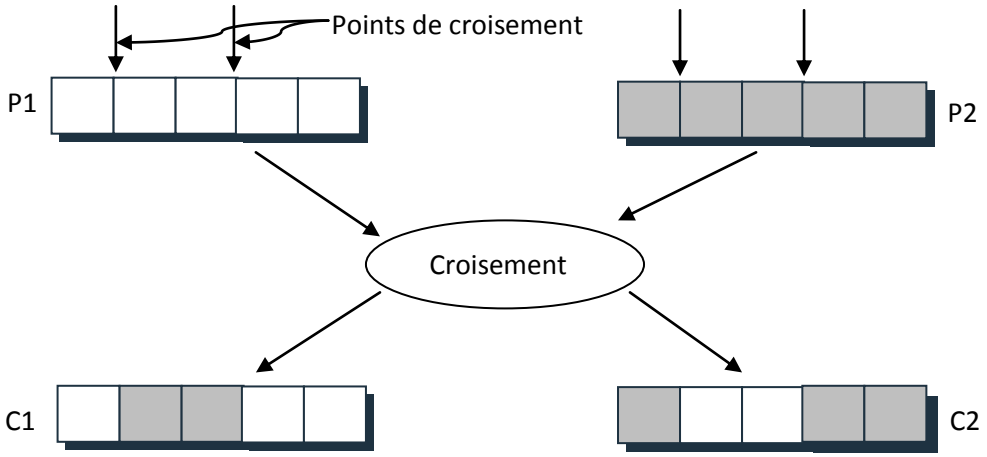


Figure 3.3.b Croisement à deux

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets (à codage binaire). Pour les problèmes continus, un croisement ``barycentrique" **[25]** peut être utilisé, où deux gènes $P_1(i)$ et $P_2(i)$ sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes $C_1(i)$ et $C_2(i)$ par combinaison linéaire :

$$\begin{cases} \boldsymbol{C_1(i) = \alpha P_1(i) + (1-\alpha) P_2(i)} \\ \boldsymbol{C_2(i) = (1-\alpha) P_1(i) + \alpha P_2(i)} \end{cases} \quad (3.2)$$

Où **α** est un coefficient de pondération aléatoire adapté au domaine d'extension des gènes (il n'est pas nécessairement compris entre 0 et 1, il peut par exemple prendre des valeurs dans l'intervalle [-0.5 ,1.5], ce qui permet de générer des points entre, ou à l'extérieur des deux gènes considérés.

3.5.3 Mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière **[23]**. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique.

La mutation est exécutée uniquement sur une seule chaîne. Elle représente la modification aléatoire (à faible probabilité) de la valeur d'un caractère de la chaîne, pour un codage binaire cela revient à changer un 1 en 0 et vice versa (figure 3.4). Cet opérateur introduit de la diversité dans le processus de recherche des solutions et peut aider l'AG à ne pas stagner dans un optimum local.

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire. Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

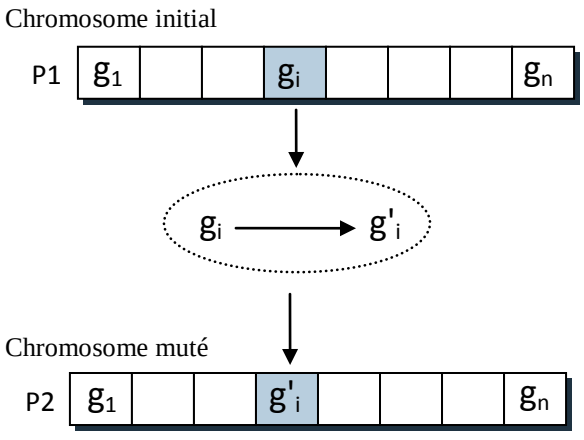


Figure 3.4: Principe de l'opérateur de mutation

Dans les problèmes continus, on procède de la même manière, en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit généralement gaussien **[27]**, (l'écart type de ce bruit est difficile à choisir a priori) ou de choisir un nouveau gène dans l'espace de recherche **[25]**.

3.6 Dynamique de codage

La première étape consiste à définir et coder convenablement le problème. A chaque variable d'optimisation x_i (à chaque paramètre du dispositif), nous faisons correspondre un *gène*. Nous appelons *chromosome* un ensemble de gènes. Chaque dispositif est représenté par un *individu* doté d'un génotype constitué d'un ou plusieurs chromosomes. Nous appelons *population* un ensemble de *N* individus que nous allons faire évoluer.
On aboutit à une structure présentant cinq niveaux d'organisation d'où résulte le comportement complexe des AG.

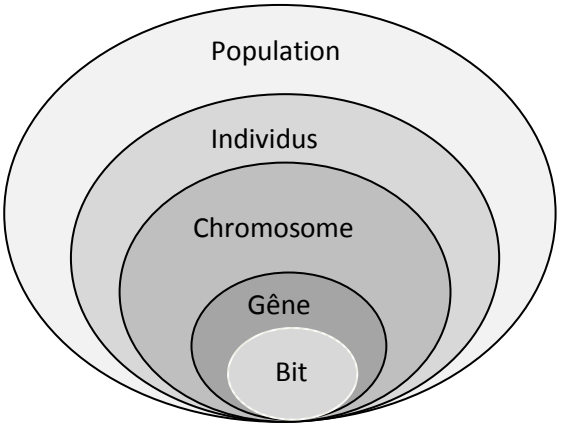


Figure 3.5 Les cinq niveaux d'organisation de notre Algorithme

Il existe plusieurs types de codage, on site ci-dessous les deux plus utilisés:

a) Codage binaire **[25]**

Dans ce type de codage, chaque individu de la population est représenté par une chaine de longueur fixe, dont les éléments (gênes) sont choisis dans un alphabet fini.
Cela concerne des problèmes combinatoires discrets, mais aussi des problèmes continus que l'on s'autorise à les discrétiser.

- ✓ Il permet de transformer un problème d'optimisation en un problème combinatoire.
- ✓ Il facilite la manipulation de tous types de variables (même pour les chaînes de caractères) en tenant compte des fonctions de codage et de décodage.

b) Codage réel **[27]**

La représentation continue, aussi dite représentation réelle, définit un algorithme génétique qui effectue sa recherche dans $\mathbb{R}^n$ ou une partie de celui-ci.

- ✓ l'adaptation aux problèmes d'optimisations numériques.
- ✓ Facilité de développer les méthodes hybrides.

3.7 Paramètres d'un algorithme génétique

Pour démarrer un A.G, il faut définir certains paramètres tel que :

- ✓ La taille de la population.
- ✓ Les probabilités de croisement et de mutation.
- ✓ Le nombre de générations.
- ✓ Le type d'opérateurs génétiques.

C'est un réglage qui doit être effectué pour chaque type de problème. Ceci constitue une part importante du travail de l'expérimentateur. Dans la littérature, ces paramètres changent d'une application à une autre et constituent un domaine de recherche qui est encore en exploration.

3.7.1 Taille de la population

L'A.G. nécessite la détermination du nombre d'individus qui constitue la population. Le problème qui se pose est : comment fixer la taille de la population ?

Une population trop petite évolue probablement vers un optimum local peut intéressant, mais avec une vitesse de convergence rapide. Une population convenable doit être choisie de façon à réaliser un compromis entre le temps de calcul et la qualité du

résultat. Plusieurs chercheurs se sont penchés sur le problème qui consiste à déterminer une taille optimale. Certains l'ont fixé entre 20 et 100 de manière empirique, d'autres essayent de l'ajuster en respectant un taux d'erreur, où la varier au cours de l'évolution de l'algorithme **[28]**.

3.7.2 Nombre de générations

C'est un chiffre que l'expérimentateur doit fixer. Il est préférable qu'il soit assez grand afin de visualiser l'espace de recherche. Certains utilisent un nombre entre 50 et 1000, d'autres jusqu'a 10000. Il dépend essentiellement du problème à solutionner, et l'essentiel est de trouver des solutions acceptables par le cahier de charge en un nombre réduit de générations.

3.7.3 Probabilité des opérateurs génétiques.

Le choix des probabilités de croisement Pc et de la probabilité de mutation Pm est un problème d'optimisation non linaire, complexe et difficile à résoudre, car ces probabilités dépendent de la nature de la fonction objective, qui elle même dépend du problème à résoudre. Cependant une formule que les auteurs considèrent est, que la probabilité Pm dépend de la taille du chromosome tel que :

$$\frac{1}{N_p} < P_m < \frac{10}{N_p} \tag{3.3}$$

Où Np est la longueur du chromosome.

Pour la plupart des problèmes standards, DeJong **[29]** conseille d'utiliser des paramètres fixés à : Np = 100, Pc = 0, 6 et Pm = $10^{-3.}$

Quant à J.D. Schaffer **[30]**, il propose une formule telle que la probabilité de mutation Pm, est en fonction de la taille de la population Np et la longueur du chromosome m.

$$P_m \approx \frac{1.75}{N_{p\sqrt{m}}} \tag{3.4}$$

Il y a aussi d'autres chercheurs qui ont proposé un réglage dynamique de ces paramètres en prenant durant les premières itérations, des probabilités de croisement et de mutation assez élevées, ce qui permet d'explorer efficacement l'espace des solutions

et par la suite, des valeurs plus faibles sont choisies, ce qui tend à stabiliser la population aux alentours des bonnes solutions. Pour ce faire, Hesser et Manner **[31],** présentent une formule en faisant intervenir le numéro **k** de la génération traitée par l'algorithme génétique:

$$p_m = p_{m0}.e^{-\alpha k/2} \qquad (3.5)$$

Pm_0 : désigne une probabilité initiale qui peut être similaire à celle de Schaffer **[30]**.

α : est une constante qui permet de régler la décroissance en fonction du temps.

Quant au choix des probabilités de croisement, des études restent toujours en cours, mais les probabilités les plus couramment utilisées appartiennent à l'intervalle [0.3 1] et rarement inférieures à 0.3.

3.8 Améliorations classiques.

3.8.1 Introduction

Les processus de sélection présentés sont très sensibles aux écarts de fitness et dans certains cas, un très bon individu risque d'être reproduit trop souvent et peut même provoquer l'élimination complète de ses congénères; on obtient alors une population homogène contenant un seul type d'individu. Ainsi, dans l'exemple de la figure (3.6), le second mode M_2 risque d'être le seul représentant pour la génération suivante et seule la mutation pourra aider à atteindre l'objectif global M_1 au prix de nombreux essais successifs **[25]**.

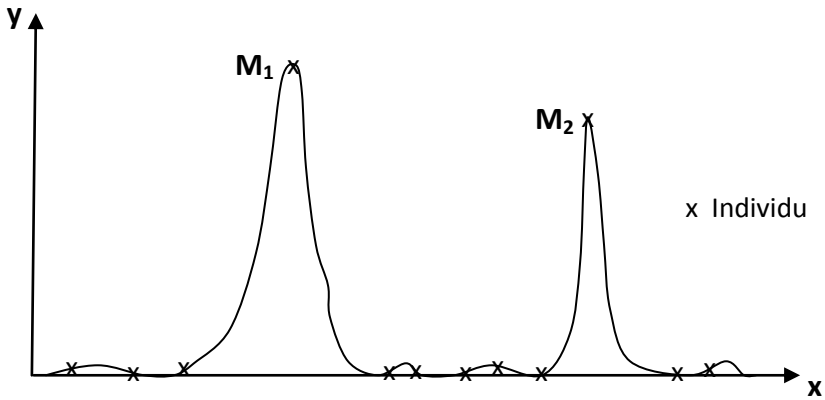


Figure 3.6: Exemple où les sélections classiques risquent de ne reproduire qu'un individu

Pour éviter ce comportement, il existe d'autres modes de sélection (*ranking*) ainsi que des principes (*scaling, sharing*) qui empêchent les individus ``forts" d'éliminer complètement les plus ``faibles". On peut également modifier le processus de sélection en introduisant des tournois entre parents et enfants, basés sur une technique proche du recuit enfin, on peut également introduire des recherches multi-objectives, en utilisant la notion de dominance lors de la sélection.

3.8.2 Mise à l'échelle (Scaling)

Le *scaling* ou mise à l'échelle, modifie les fitness afin de réduire ou d'amplifier artificiellement les écarts entre les individus. Le processus de sélection n'opère plus sur la fitness réelle mais sur son image après scaling. Parmi les fonctions de scaling, on peut envisager le scaling linéaire et le scaling exponentiel. Soit f_r la fitness avant scaling et f_s la fitness modifiée par le scaling **[25]**.

- Scaling linéaire:

Dans ce cas la fonction de scaling est définie de la façon suivante **[16]** :

$$f_s = af_r + b$$

$$a = \frac{max' - min'}{max - min} \quad ; \quad b = \frac{min'.max - min.max'}{max - min}$$

En règle générale, le coefficient *a* est inférieur à un, ce qui permet de réduire les écarts de fitness et donc de favoriser l'exploration de l'espace. Ce scaling est statique par rapport au numéro de génération et pénalise la fin de convergence lorsque l'on désire favoriser les modes dominants.

- Scaling exponentiel:

Il est défini de la façon suivante **[25]** (voir figure 3.7):

$$f_s = (f_r)^{k(n)}$$

Où *n* est la génération courante.

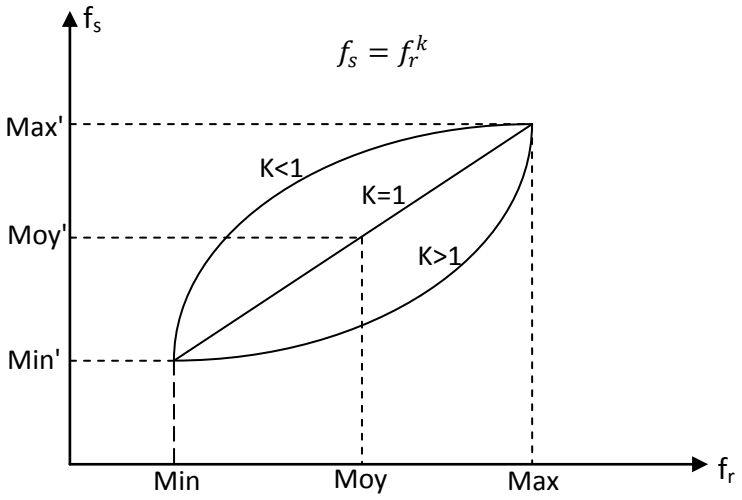


Figure 3.7: Fonction de scaling exponentielle

- ✓ Pour *k* proche de zéro, on réduit fortement les écarts de fitness ; aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un algorithme de recherche aléatoire et permet d'explorer l'espace.
- ✓ Pour *k* proche de 1 : le scaling est inopérant.
- ✓ Pour $k > 1$ les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

3.8.3 Sharing

L'objectif du sharing est de répartir sur chaque sommet de la fonction à optimiser un nombre d'individus proportionnel à la fitness associée à ce sommet **[25]**. De la même façon que le scaling, le sharing consiste à modifier la fitness utilisée par le processus de sélection. Pour éviter le rassemblement des individus autour d'un sommet dominant, le sharing pénalise les fitness en fonction du taux d'agrégation de la population dans le voisinage d'un individu. Il requiert l'introduction d'une notion de distance. Dans la pratique, il faut définir une distance indiquant la dissimilarité entre deux individus. Cette distance est alors utilisée pour calculer la nouvelle fitness de la façon suivante :

$$f_i = \frac{f_i}{m_i'} \qquad ; \quad m_i' = \sum_{j=1}^{N} S(d(x_i, x_j))$$

Avec

$$\begin{cases} S(d) = 1 - \left(\frac{d}{\sigma_{share}}\right) \propto & si \quad d < \sigma_{share} \\ S(d) = 0 & si \quad d > \sigma_{share} \end{cases}$$

3.9 Algorithmes génétiques hiérarchiques

3.9.1 Introduction

Dans le souci de développer et de résoudre des problèmes complexes, les chercheurs se sont penchés profondément sur la structure interne de l'ADN et ils ont remarqué que la structure génétique du chromosome subit un certain nombre de changements au niveau de leurs gènes, et qu'ils sont arrangés de façon hiérarchique **[32]**.

Aussi certains gènes dominent d'autres gènes, et qu'il y a aussi des gènes actifs et d'autres inactifs. Un tel phénomène a incité les chercheurs à imiter la structure de l'ADN de façon à ce qu'on puisse avoir une structure génétique plus complexe mais évolutive.

3..9.2 Structure hiérarchique de l'ADN

L'information génétique est localisée au niveau d'une substance visqueuse identifiée comme étant l'acide désoxyribonucléique (ADN). Elle est servie comme un livre de recettes pour construire toutes les protéines nécessaires au bon fonctionnement des cellules.

Un gène est un bout d'ADN qui contient les informations nécessaires de la construction d'une protéine spécifique. Dans un noyau de cellule humaine, il y aurait 100000 recettes (information) différentes.

En réalité il existe au niveau de la chaîne d'ADN des gènes qui ne sont pas activé, ceci a été montré en 1961 par Jacob et Monod, ils ont établi un modèle pour lequel on distingue deux types des gènes :

Les gènes qui s'occupent de la synthèse des protéines sont appelés ***gènes de structure*** (ou *régulateurs de séquence*).il sont codées en polypeptide (ARNs).

Chaque cellule contient tous les chromosomes, et donc tous les gènes d'un être vivant. Dans ce cas, comment se fait-il que les cellules musculaires soient différentes des neurones?

Les ***gènes régulateurs*** régissent l'expression des gènes de structure. Cela signifie que ces gènes ont comme fonction de décider si un gène s'exprime ou non. L'ensemble des gènes régulateurs qui se trouve dans l'ADN est appelé ***promoteur.***

3.9.3 Structure hiérarchique du chromosome

Par analogie avec la structure de l'ADN, une structure multi-niveaux de contrôle peut être développée à l'intérieur du chromosome, ce qui permet d'introduire une nouvelle structure au niveau des AG. Ceci consiste à activer un gène à l'aide d'un gène de contrôle de premier niveau. Une fois cette valeur est activée, celle ci constituera un deuxième niveau de contrôle et ainsi de suite jusqu'à la configuration totale des gènes de structure (voir figure 3.8). Par analogie avec cette inspiration, les gènes de contrôle sont représentés en binaires, et les gènes de structure peuvent être représentés en binaires, réels ou les deux à la fois.

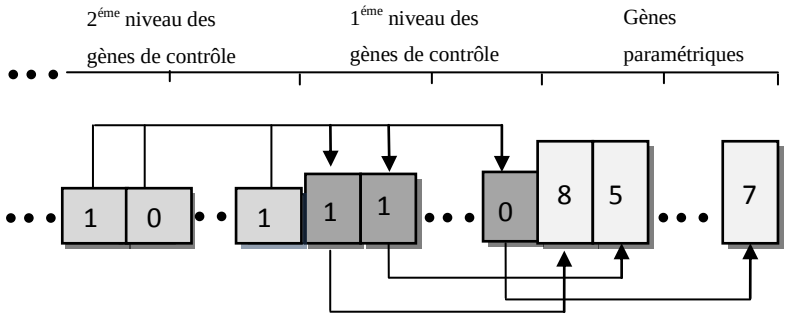


Figure 3.8 : structure hiérarchique du chromosome

De ce fait, on a une grande souplesse pour gouverner tout le chromosome ou même une partie du chromosome pour effectuer une tâche bien déterminée. Plusieurs niveaux peuvent être utilisés en fonction du problème posé.

Pour mieux comprendre l'action des gènes de contrôle, on présentera deux exemples illustrés par les figures 3.9.a et 3.9.b, qui représentent les phénotypes C_1 et C_2 actifs, issus de la transcription du promoteur RS.

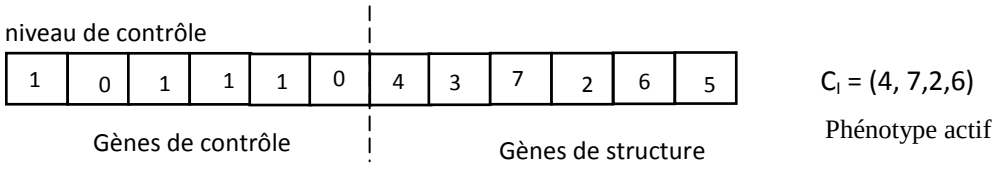


Figure 3.9.a : Cas d'un chromosome à un niveau de contrôle.

L'état ' 1 ' du bit de contrôle indique l'activation du gène et '0' sa désactivation

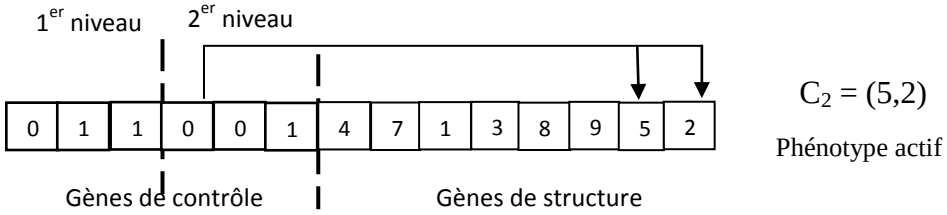


Figure 3.9.b : Cas d'un chromosome à deux niveaux de contrôle.

Les deux chromosomes C_1 et C_2 sont de longueurs respectivement 4 et 2, ce qui veut dire que cette structure nous permette de générer des phénotypes de longueurs différentes, et par conséquent des chromosomes à taille variable. La structure hiérarchique permet de trouver une solution considérant toutes les longueurs possibles (structures différentes) et toutes les valeurs des paramètres pour atteindre les critères de la fonction objective.

Plusieurs niveaux hiérarchiques peuvent être utilisés au besoin, la figure (3.10) illustre un exemple de structure à 3 niveaux hiérarchiques.

Exemple :

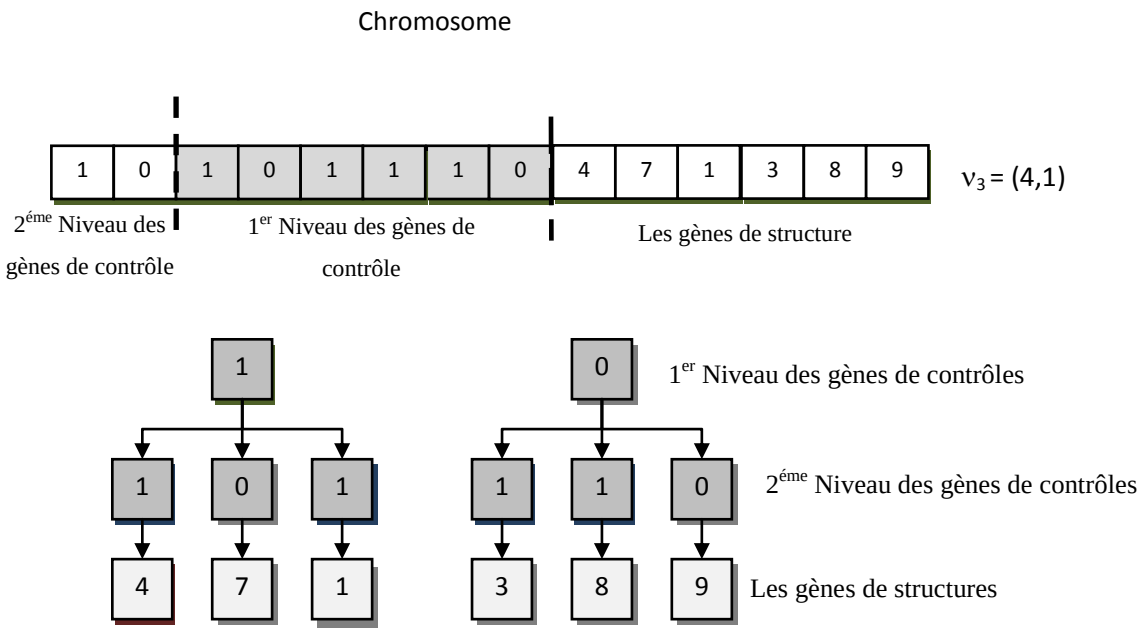


Figure 3.10 : structure d'un chromosome à 3 niveaux hiérarchiques

3.9.4 Cycle génétique d'un HGA.

Après avoir donné la structure d'un chromosome hiérarchique, on passera maintenant au cycle génétique qui n'est autre que l'organigramme de fonctionnement général d'un H.G.A (terme anglo saxon : Hierarchical Genetic Algorithms) illustré par la figure 3.11, et qui servira comme élément de base pour la réalisation de notre programme de simulation sous MATLAB. La population initiale est obtenue en générant aléatoirement des chaînes de chromosomes afin de démarrer avec des solutions les plus diversifiées possibles.

La structure hiérarchique permet de trouver une solution considérant toutes les longueurs possibles (structures différentes) et toutes les valeurs de paramètres pour rencontrer les critères de la fonction objective.

Dans le présent cas, la population est constituée de deux sous populations X et Y, tel que X représente la sous population des gènes de contrôle, codée en binaire et Y la sous population des gènes de structure codée en réelle.

3.9.5 Opérateurs génétiques:

D'après l'organigramme de la figure (3.11), l'opérateur de sélection est appliqué sur toute la population globale (X et Y) afin de donner le meilleur chromosome. Cette opération est accomplie en utilisant une fonction objective dépendante du problème ou du cahier de charge choisi.

Quant aux autres opérateurs génétiques, ils sont appliqués séparément sur les gênes de contrôle et les gênes de structure auxquels sera appliqué les opérateurs de croisement et de mutation décrit dans §3.5.3. Les méthodes standards de croisement et de mutation peuvent être utilisées indépendamment pour chaque niveau de gènes (contrôle ou structure), ou sur le chromosome en entier s'il est homogène.

Les opérations génétiques qui affectent les gênes des niveaux supérieurs peuvent éventuellement affecter le nombre et la topologie du système à optimiser.

a) Population:

Elle peut être obtenue en générant aléatoirement les chaines de l'espace de recherche, cette méthode a l'avantage de commencer la recherche à partir de diverses solutions de l'espace de recherche.

La population globale est constituée par deux sous populations X, Y, une sous population (X) formée par des chaines codées en binaire, et représentons les gênes de contrôle. La deuxième (Y) représente les gênes de structure (coefficients de filtre par exemple).

b) Probabilité de croisement et de mutation

L'ordre de grandeur des probabilités est sensiblement le même que pour les A.G, sauf que les probabilités de gènes de contrôle sont généralement légèrement inférieures à celles des gènes de structure **[25]**.

Les méthodes standard de croisement et de mutation peuvent être utilisées indépendamment pour chaque niveau de gênes, ou sur tout de chromosome s'il est homogène.

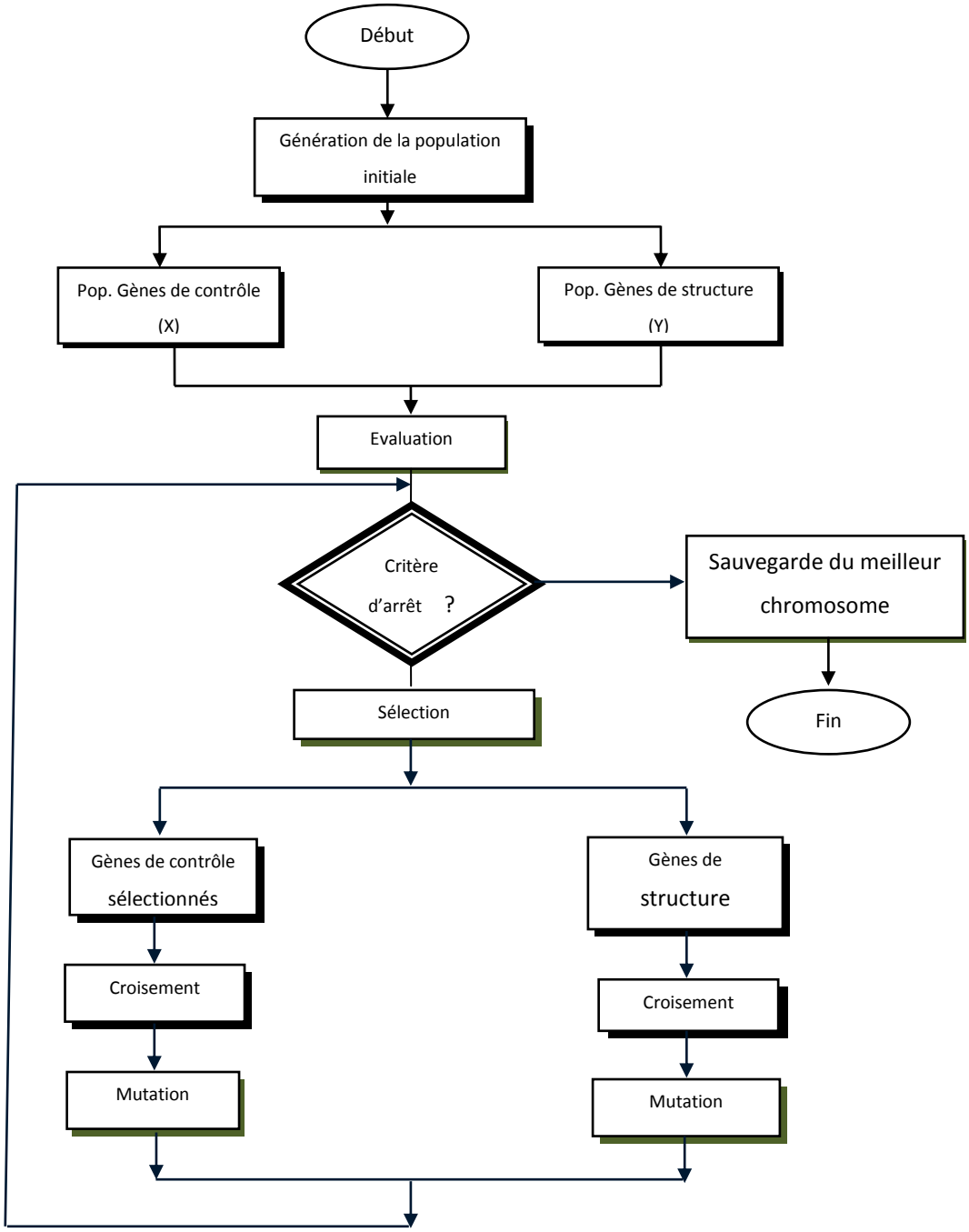


Figure 3.11 : Organigramme d'évolution d'un H.G.A.

3.10 Avantages des algorithmes génétiques

Contrairement à la recherche opérationnelle, l'AG n'exige aucune connaissance de la manière à résoudre le problème, il est seulement nécessaire de pouvoir évaluer la qualité de la solution, également, dans le cas de recherche d'optimum de fonctions analytiques, la dérivabilité et la continuité ne sont pas nécessaires.

Dans la table suivante on compare l'efficacité de l'approche classique par rapport à l'approche génétique.

Table 3.2 Comparaison entre recherche opérationnelle et recherche génétique

	Approche opérationnelle	**Approche génétique**
Rapidité	Selon solution, parfois bonne	Bonne, à très bonne
Performance	Selon solution	Bonne
Compréhension du problème	Nécessaire	Non nécessaire
Travail humain	Très variable	Faible
Applicabilité	Faible	Générale

3.11 Conclusion

Dans ce chapitre on a pu voir un aperçu général sur les concepts des AG. En conclusion, bien que leur conception soit relativement simple, les AG peuvent résoudre des problèmes assez complexes. La résolution de ces derniers est obtenue grâce aux opérateurs génétiques. Les AG sont des procédures assez robustes pour résoudre des problèmes d'optimisation **[24]**, néanmoins elles présentent certaines limites et des difficultés. Ces difficultés résident sur le choix des bons paramètres tels que : la taille de la population, le nombre de générations, les probabilités de croisement et de mutation et le choix des méthodes des opérateurs génétiques. Ceux-ci dépendent essentiellement du problème à résoudre et de sa codification.

CHAPITRE 4

SYNTHESE DE FILTRES NUMERIQUES PAR ALGORITMES GENETIQUES

4.1 Introduction

La conception de filtres numériques consiste à déterminer les coefficients et l'ordre de la fonction de transfert du filtre. Des schémas de conception classiques démarrent d'un prototype de filtre passe-bas analogique qui satisfait les spécifications prescrites, puis convertissent le filtre prototype en un filtre numérique par des transformations algébriques appropriées **[5][7]**. Parmi ces transformations, la transformation bilinéaire a longtemps constitué un bon choix car elle préserve la réponse fréquentielle et les contraintes de stabilité du filtre original sans l'introduction du problème de chevauchement. Autres types de filtres tels que, filtre passe-haut (FPH), filtre passe-bande (FPB), et filtre coupe-bande (FCB) peuvent être obtenus à partir du prototype passe-bas (FPb) par d'autres transformations algébriques **[14]**.

La conception de filtre numérique est en fait un problème d'optimisation, qui consiste à trouver une fonction rationnelle, qui soit la meilleure approximation de la fonction donnée (spécification du filtre). L'erreur définie par la différence entre la réponse impulsionnelle désirée et la réponse impulsionnelle actuelle possède généralement plusieurs objectifs en fonction des coefficients du filtre. Dans les problèmes d'optimisation à multi-objectifs, les algorithmes analytiques à base de gradient peuvent se piéger dans un minimum local (figure 4.1), et souvent ne convergent pas vers l'optimum global **[22][24]**.

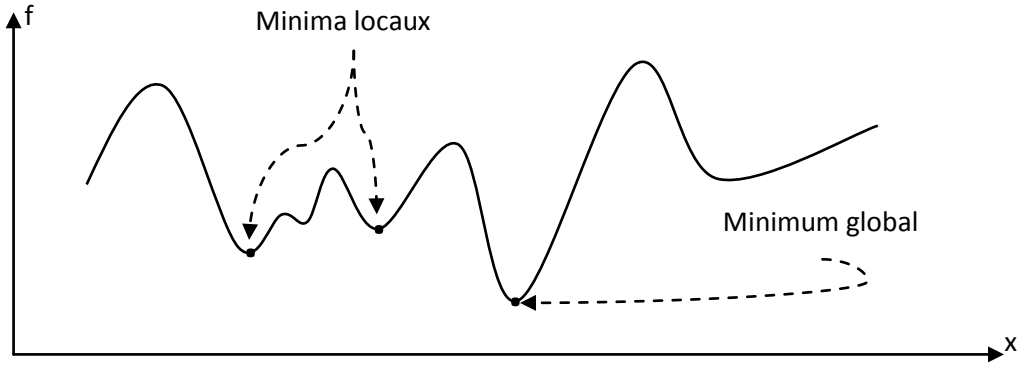


Figure 4.1 Représentation d'exemples de minimums locaux et minimum global d'une fonction f

Les méthodes de recherche à base de population aléatoire, tels que les algorithmes génétiques ont été très bénéfiques dans la recherche de l'optimum global dans les problèmes d'optimisation à multi-objectifs. Parmi les avantages clefs des méthodes de conception de filtres à base d'AG, nous citons l'implémentation de filtres numériques sans l'utilisation de transformations algébriques à partir de filtres analogiques **[5]**. Autres types de filtres tels que FPH, FPB, FCB, peuvent être conçus indépendamment, sans l'utilisation de transformations à partir de filtre canonique **[8]**.

4.2 Stratégie d'optimisation d'un filtre RII par AG :

Comme toutes les applications d'algorithme génétique, la conception de filtres numériques par AG nécessite la définition de certains paramètres nécessaires pour la configuration de l'algorithme **[33][34]**, tels que :

- La population initiale,
- Le codage des chromosomes,
- La fonction d'évaluation,
- Les opérateurs génétiques.

Dans le présent travail, nous avons utilisé deux types d'algorithmes génétiques, AGS (algorithmes génétiques standards) et AGH (algorithmes génétiques hiérarchiques), pour cela on présentera pour chaque type d'AG ses propres paramètres.

4.2.1 Algorithme génétique standard :

4.2.1.1 Génération de la population initiale:

Le premier pas dans l'implantation des algorithmes génétiques consiste à créer une population d'individus initiaux. En effet, les algorithmes génétiques agissent sur une population d'individus, et non pas sur un individu isolé.

Dans notre cas, on doit générer aléatoirement un nombre Npop de chromosomes (individus) totalement indépendants. Le nombre Npop de chromosomes par population agit surtout sur la vitesse de convergence de l'algorithme, il peut être choisi après plusieurs tests.

4.2.1.2 Codage des chromosomes:

On a vu au chapitre 3, que les chromosomes représentent des solutions potentielles au problème étudié, par conséquent dans notre cas, le chromosome doit représenter une solution de filtre numérique. Plusieurs façons sont envisageables pour représenter un filtre, on cite par exemple **[35][36]** :

- Les coefficients de la fonction de transfert rationnelle (structure directe),
- Les pôles et les zéros,
- Les coefficients de cellules de 1^{ér} et 2^{éme}ordre (structure cascade ou parallèle).

Comme il est montré à la figure 4.2, les gènes du chromosome représentent les coefficients des cellules du 1^{ér} et du 2^{éme} ordre, constituant la fonction de transfert de notre filtre, celle-ci peut être implémentée sous plusieurs formes (structures) par produit (structure cascade) ou par sommation (structure parallèle).

Dans notre cas, une réalisation en cascade de cellules du second ordre et du premier ordre à été adoptée pour représenter le modèle du filtre numérique RII.

La fonction de transfert du filtre numérique représentée en une cascade de termes du 1^{ér} et 2^{éme} ordre est donnée par l'équation ci-dessous :

$$H(z) = g \prod_{i=1}^{M} \left(\frac{1+b_i z^{-1}}{1+a_i z^{-1}}\right) \prod_{j=1}^{N} \left(\frac{b_{j0}+b_{j1} z^{-1}+b_{j2} z^{-2}}{1+a_{j1} z^{-1}+a_{j2} z^{-2}}\right) \quad (4.1)$$

La variable « g » étant le gain du filtre, les coefficients du dénominateur a_i, a_{j1}, et a_{j2} sont associés aux pôles, quant à b_i, b_{j0}, b_{j1}, et b_{j2}, ils représentent les coefficients du numérateur associés au zéros. Les variables M et N dénotent respectivement, le nombre de termes du 1^{ér} ordre et le nombre de termes du 2^{éme} ordre. D'après ce modèle, la fonction de transfert du filtre contient donc M termes du 1^{ér} ordre et N termes du 2^{éme} ordre, et par conséquent, l'ordre global du filtre peut être calculé comme suit :

Ordre du filtre = **M** x (ordre d'une cellule du 1^{ér} ordre) + **N** x (ordre d'une cellule du 2^{éme} ordre)

Ordre du filtre = M + 2 N

M et N sont choisis par l'utilisateur, pour définir l'ordre et la structure du filtre voulu.

Un chromosome est composé d'une série de gènes, chaque gène correspond au coefficient du filtre des termes du 1^{ér} et 2^{éme} ordre de la fonction de transfert. L'ordre du filtre détermine le nombre total de gènes dans un chromosome, la figure 4.2 représente la structure du chromosome.

Le chromosome est subdivisé en 03 différentes parties (C1, C2 et C3).

- C1 : pour les coefficients des cellules du 1^{ér} ordre ;
- C2 : contient les coefficients du 2^{éme} ordre ;
- C3 : pour le gène représentant le gain g.

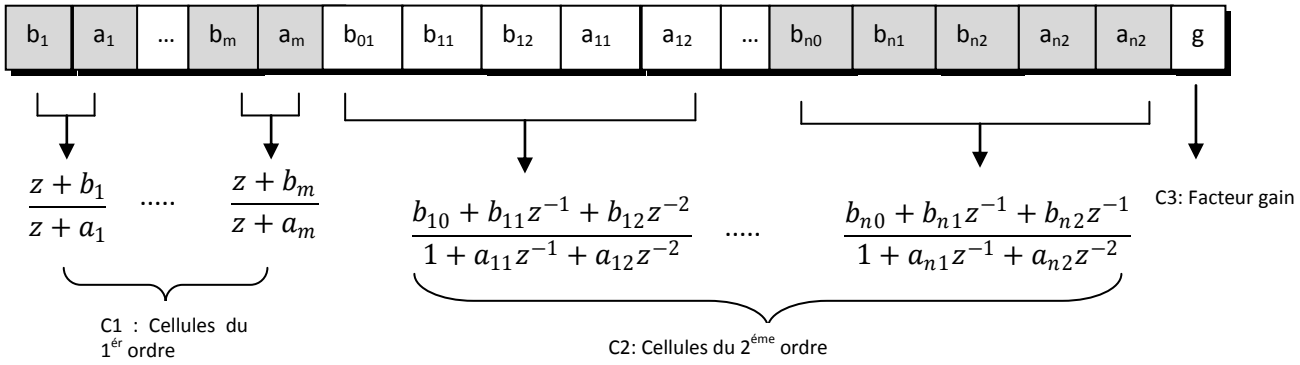


Figure 4.2 Représentation d'un chromosome de coefficients du filtre RII en AGS

La partie C1 contient les doublets (b_i, a_i) représentant les cellules du 1^{ér} ordre, et la partie C2 est constituée des ensembles de coefficients (b_{0i}, b_{1i}, b_{2i}, a_{1i}, a_{2i}) représentant les cellules du 2^{éme} ordre (figure 4.2).

Etant donné, l'ordre maximal d'un filtre RII (M+2N), il existe différentes combinaisons de termes du 1^{ér} et 2^{éme} ordre pour construire un filtre. Par exemple, un filtre du 5^{éme} ordre peut avoir trois combinaisons de termes ; 2 termes du 2^{éme} ordre et 1 terme du 1^{ér} ordre, 1 terme du 2^{éme} ordre et 3 termes du 1^{ér} ordre et 05 termes du 1^{ér} ordre.

4.2.1.2 Fonction d'évaluation**: [36][37]**

La conception d'un filtre numérique est souvent basée sur l'optimisation de la réponse en amplitude dans le domaine fréquentiel.

Les spécifications typiques de filtre englobent plusieurs paramètres de conception, tels que la fréquence de coupure, les ondulations en passe-bande, la largeur de la bande de transition etc.…, la figure 4.3 représente un schéma bloc qui montre un synoptique de conception général de filtre RII utilisant les algorithmes génétiques.

Un filtre numérique RII ayant une fonction de transfert H(z), sous la forme de fonction rationnelle, de la variable complexe z^{-1}, a une réponse en fréquence H ($e^{j\omega}$).

La différence ε(ω) entre la réponse en fréquence du filtre H(z) à optimiser et la spécification donnée du filtre $H_d(z)$ est définie par:

$$\varepsilon(\omega) = \left|H_d(e^{j\omega})\right| - \left|H(e^{j\omega})\right| \tag{4.2}$$

La réponse en amplitude $\left|H_d(e^{j\omega})\right|$ représente la spécification du filtre desirée, et $\left|H(e^{j\omega})\right|$ indique la réponse en fréquence du filtre actuel. L'algorithme génétique cherche les coefficients du filtre optimal en maximisant la fonction d'évaluation (f_{eval}) définie comme l'inverse de la fonction d'erreur **[35]**.

$$f_{eval} = {}^{1}/_{\varepsilon(\omega)} \tag{4.3}$$

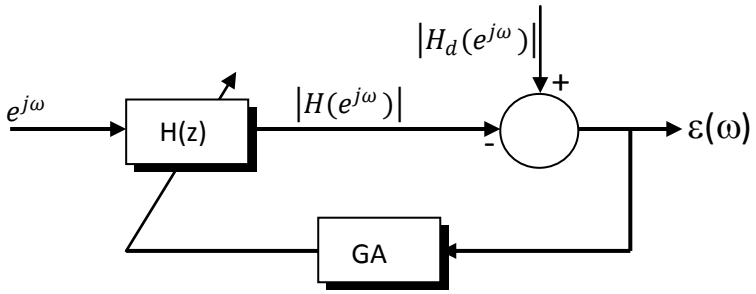


Figure 4.3 : conception de filtre numérique RII utilisant les algorithmes génétiques

La figure 4.4 représente graphiquement l'erreur dans le domaine fréquentiel produite par un filtre numérique RII à optimisé. La différence entre l'amplitude de la réponse actuelle du filtre et les spécifications désirées du filtre définissent l'erreur à minimiser. Si la réponse du filtre satisfait les spécifications, donc aucune erreur ne s'est produite.

On considère la plage de fréquence de 0 à la fréquence d'échantillonnage normalisée (π).

La région "C" (figure (4.4)), indique la plage de fréquence où la réponse du filtre ne réponds pas aux spécifications désirées, l'erreur dans le domaine fréquentiel est utilisée pour définir la fonction d'évaluation de l'algorithme génétique. La fonction d'évaluation f est l'inverse de la somme des erreurs quadratiques de la bande passante et la bande coupée dans la région C.

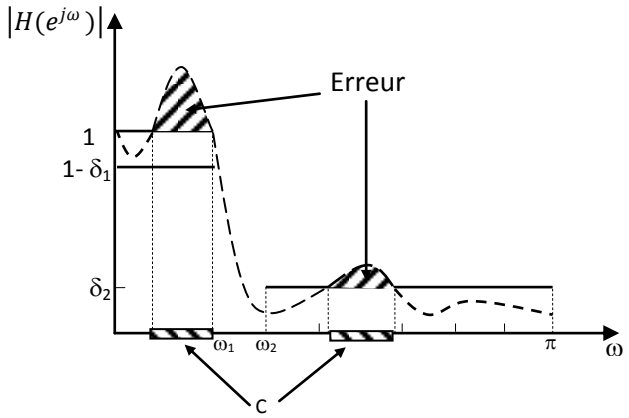


Figure 4.4 : L'erreur dans la bande passante entre la réponse actuelle et les spécifications désirées du filtre

$$f_{eval} = \frac{1}{1+\sum_{\omega \epsilon C}\left[\varepsilon_p^2(\omega)+\varepsilon_s^2(\omega)\right]} \tag{4.4}$$

$$\begin{cases} \varepsilon_p(\omega) = \begin{cases} |H(e^{j\omega})| - 1 & si\ |H(e^{j\omega})| > 1 \\ (1-\delta_1) - |H(e^{j\omega})| & si\ |H(e^{j\omega})| < (1-\delta_1) \end{cases} \\ \varepsilon_s(\omega) = |H(e^{j\omega})| - \delta_2 \qquad |H(e^{j\omega})| > \delta_2 \end{cases} \tag{4.5}$$

Ou $\varepsilon_p(\omega)$ et $\varepsilon_s(\omega)$ désignent les erreurs entre la réponse actuelle et la réponse désirée dans la bande passante et la bande coupée respectivement. L'erreur totale est la somme des erreurs de la bande passante et la bande coupée. Le paramètre δ_1 définie l'ondulation dans la bande passante et δ_2 indique le gain de la bande coupée, ils sont donnés dans les spécifications du filtre.

Cette fonction d'évaluation ou "fitness" constitue la pièce la plus importante pour un algorithme génétique, chaque application possède sa propre fonction d'évaluation, et le bon

choix de cette dernière, favorise pleinement la convergence sûre et rapide vers un optimum global.

Pour la synthèse de filtres numériques RII, et en particulier dans notre travail, nous avons opté pour une fonction a multi objectifs, autrement dit, plusieurs contraintes ont été prises en considération pour affiner le processus de recherche.

Les critères pris en considération pour un AGS sont:

1- **L'erreur** : Calculée entre la réponse fréquentielle désirée (gabarit) et la réponse fréquentielle actuelle, elle est définie par les équations 4.4 et 4.5 ;

2- **Le nombre de dépassement** : Nombre d'échantillons de la réponse en fréquence situés hors intervalle autorisé [$1+\delta_1$, $1-\delta_2$] dans la bande passante, un exemple est illustré par la figure 4.5.

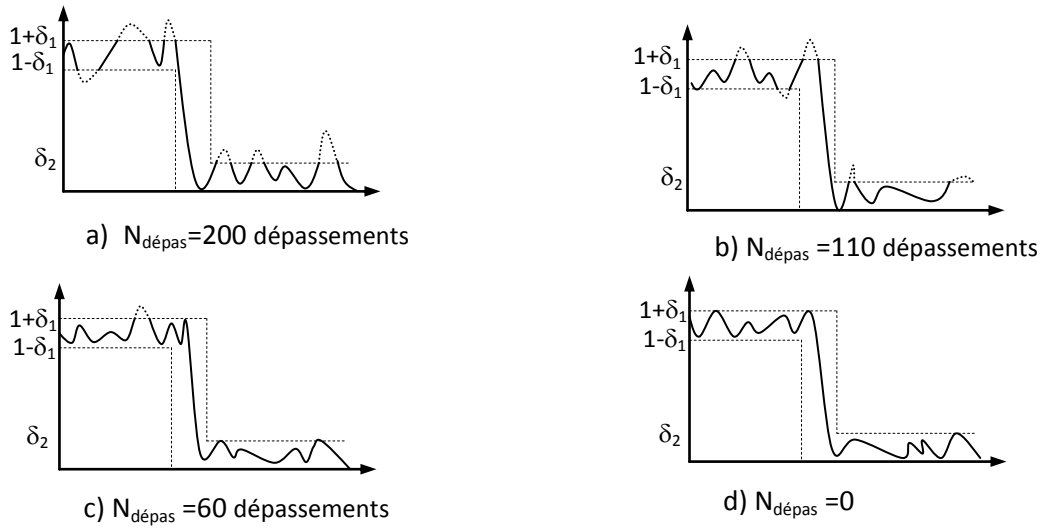


Figure 4.5 Exemple de nombre de dépassement, a) N=200, b) N=110, c= N=60 et d) N=0

Plus le nombre de dépassements ($N_{dépas}$) est élevé (figure 4.5), plus la valeur de la fonction d'évaluation attribuée au filtre est petite, et plus le filtre est mauvais.

3- **Critère de stabilité :** A chaque génération, la stabilité des filtres est testée (position des pôles), et les filtres instables sont pénalisés en leur attribuant une valeur de la fonction d'évaluation faible, moyennant le facteur de stabilité défini comme suit :

$$f_{stab} = \begin{cases} 0 & si\ filtre\ stable \\ 1000 & si\ filtre\ instable \end{cases} \quad (4.6)$$

Et en vertu de ces trois facteurs, la fonction d'évaluation peut être définie comme ci-dessous :

$$f_{éval} = {}^{1}/_{(1 + w_1.\varepsilon_p + w_2.\varepsilon_s + N_{dépas} + f_{stab})} \quad (4.7)$$

Les coefficients de pondération w_1 et w_2 permettent d'affiner la recherche et la convergence vers l'optimum global.

4.2.1.3 Opérateurs génétiques

L'optimisation par les algorithmes génétiques est réalisée en utilisant l'évaluation, la sélection, et les opérateurs génétiques. La caractéristique de stabilité étant très importante dans le domaine de filtrage numérique, et dans le but d'orienter le processus de la recherche génétique vers une solution (filtre) stable, tous les chromosomes dans la population initiale, sont choisis pour satisfaire les contraintes de stabilité (chapitre 1).

Comme on l'a vu au chapitre (3), trois opérateurs sont nécessaires à l'élaboration d'un algorithme génétique, à savoir, la sélection, le croisement et la mutation.

a) Sélection:

Le choix le plus populaire pour l'opérateur de sélection est la méthode de roulette (Wheel sélection) qui sélectionne des individus avec une probabilité proportionnelle à la valeur de leurs évaluations. Les individus ayant des valeurs d'évaluation élevées ont plus de chance d'être sélectionnés pour les opérations génétiques suivantes, la stabilité n'est pas affectée à ce niveau.

Les opérations génétiques doivent être choisies de façon à produire des coefficients de filtre qui assurent la stabilité du filtre RII. Dans le but de garantir que le filtre RII réalisé soit stable, les pôles associés aux coefficients du polynôme dénominateur doivent être à l'intérieur du cercle unité du plan complexe. Les opérations de croisement et de mutation permettent de produire de nouveaux coefficients.

b) Croisement:

Pour l'application d'algorithmes génétiques standards AGS à la synthèse de filtres numériques, on a utilisé le type de croisement classique nommé aussi « croisement à 1 point ». Ce croisement consiste, qu'à partir de deux individus, on obtient deux nouveaux individus (enfants) qui héritent de certaines caractéristiques de leurs parents. Les individus parents sont coupés au niveau d'un point appelé point de croisement figure (3.3.a)., choisi aléatoirement.

c) Mutation:

L'opération de mutation produit des variations aléatoires au niveau des coefficients, pour une recherche génétique effective. Le but de l'opérateur de mutation consiste à éviter à l'AG de converger vers des extremums locaux de la fonction d'évaluation, et de permettre de créer des éléments originaux. Le type de mutation choisi pour le cas de l'AGS, consiste à remplacer le gène actuel par un autre, pris aléatoirement dans l'espace de recherche, et ceci avec une certaine probabilité p_m qui représente la fréquence à laquelle les gènes d'un chromosome seront mutés.

4.2.2 Algorithme génétique hiérarchique:

4.2.2.1 Structure des chromosomes :

Comme on l'a vu au chapitre 3, le principal atout des algorithmes génétiques hiérarchiques consiste à coder les paramètres d'un système en une structure hiérarchique. Le codage du chromosome pour un AGH, est légèrement différent de celui cité pour un AGS, et pour comprendre ce codage, un exemple d'application est illustré ci-dessous.

Soit H(z) la forme générale de la fonction de transfert du filtre à concevoir, la représentation du chromosome correspondant est donnée à la figure 4.6.

$$H(z) = |g| \frac{(z+b_1)(z+b_2)(z^2+b_{11}z+b_{12})(z^2+b_{21}z+b_{22})}{(z+a_1)(z+a_2)(z^2+a_{11}z+a_{21})(z^2+a_{12}z+a_{22})} \quad (4.8)$$

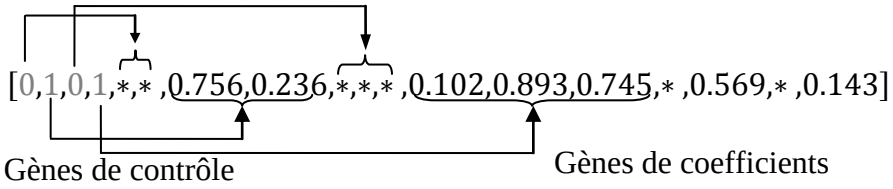
Où k désigne le gain, et a_i, a_{j1}, a_{j2}, b_i, b_{j1}, b_{j2} pour i =1,2 et j=1,2 les coefficients.
Dans cette configuration, il existe deux types de gènes qui représente notre fonction de transfert H(z) ; gène de contrôle et gène de coefficient.

Les gènes de contrôle étant choisi en format binaire (0 ou 1), décide de l'état d'activation de chaque block (cellule du filtre). Un « 1 » signifie l'état d'activation, et « 0 » représente l'état désactivé. Le gène coefficient (qui peut être codé en binaire, réel ou autre) définie la valeur des coefficients dans chaque block (cellule), La formulation peut être illustrée par l'exemple suivant.

Considérons la fonction de transfert suivante :

$$H(z) = 1.12 * \frac{(1+0.756z^{-1})(0.102+0.893z^{-1}+0.745z^{-2})}{(1+0.236z^{-1})(1+0.569z^{-1}+0143z^{-2})} \quad (4.9)$$

La formulation des gènes pour un simple chromosome, comme indiqué dans la figure (4.6) est donc :



La figure 4.6 représente un cas plus général avec 2 cellules du 1^{ér} ordre (M=2), et 2 cellules du 2^{éme} ordre (N=2).

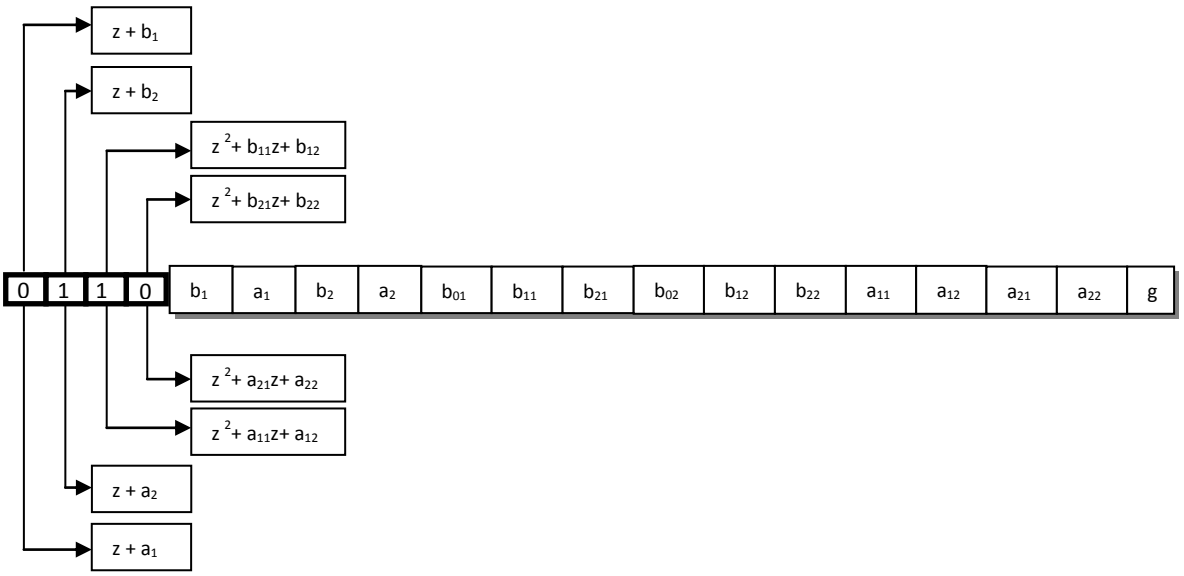


Figure 4.6 Structure d'un chromosome pour un algorithme génétique hiérarchique

La taille des chromosomes dépend de la nature de la structure utilisée, pour cette raison on définie deux constants *N* et *M* qui représentent respectivement le nombre de cellules de premier ordre et celui du second ordre. (Pour notre exemple on a *N=2* et *M=2*).

Les gènes de contrôle sont des chaînes codées en binaire, ils permettent de commander l'activation de chaque cellule de la structure, donc la taille de la chaîne de gènes de contrôle est égale au nombre de cellules ;

Notons : $T_1 = (N+M) = 4$, T_1 représente la taille de la chaîne de gènes de contrôle

Les gènes de coefficients représentent les coefficients a_i, b_i, a_{ij}, b_{ij} et g du filtre, d'où la taille de la chaine des gènes de coefficients, $T_2 = 2N+5M+1 = 14$.

Donc la taille totale du chromosome $T = T_1 + T_2 = 3N + 5M + 1 = 4 + 14 = 18$.

Dans la théorie, chaque chromosome peut être considéré comme une solution potentielle de la fonction de transfert H(z) , pour cela il ne faut pas oublier que le choix des coefficients obéit aux conditions de stabilité, cependant, et puisque le filtre est constitué d'une combinaison de cellules de premier et second ordre, la stabilité d'un tel filtre est réalisée lorsque les pôles des cellules de premier ordre se trouvent à l'intérieur du cercle unité, c.-à-d. les coefficients correspondants appartiennent à l'intervalle [-1 1], et les coefficients du dénominateur des cellules de second ordre $(z^2 + a_{1i} z + a_{2i})$ se trouvent à l'intérieur du triangle de stabilité (figure 1.10).

Donc, le choix des coefficients des cellules de deuxième ordre se fait de la manière suivante :

$$-1 \le a_{2i} \le 1 \qquad \text{et} \qquad -1 - a_{2i} \le a_{1i} \le 1 + a_{2i} \tag{4.10}$$

Pour toute la population, chaque chromosome est défini comme suit : 4 valeurs codées en binaire pour les gènes de contrôle, 12 valeurs codées en réels et comprises entre -1 et 1, définissent les coefficients a_i, b_i, a_{i2}, b_{i2} et k pour la première chaîne des gènes de structure.

Et finalement, 2 valeurs codées en réels, et dépendants de la première chaîne de gènes de coefficients (compris entre $-1 - a_{2i}$ et $1 + a_{2i}$ pour les a_{1i} et entre $-1 - b_{2i}$ et $1 + b_{2i}$ pour les b_{1i}), pour le reste des gènes de structure **[37][38]**.

4.2.2.2 L'évaluation des individus :

Pour évaluer les individus dans le cas des algorithmes génétiques hiérarchiques, on considère les mêmes critères cités pour le cas des AGs, auxquels on a ajouté une autre contrainte spécifique aux AGH, qui consiste à l'optimisation de l'ordre du filtre, en favorisant les structures les moins compliquées (ordre faible).

Dans la chaine contenant les gènes de contrôles, si on considère que la variable "O_{rd}" représente le nombre de zéros « 0 », par conséquent, plus le nombre de zéros augmente plus l'ordre du filtre est faible:

" O_{rd} " ↗ $\Rightarrow$ *Ordre du filtre plus faible*

0	1	1	0	.....	0

Figure 4.7: Exemple d'une Chaine de gènes de contrôle

Notre but consiste à attribuer aux solutions ayant une valeur de la variable " O_{rd} " élevée, une valeur de fitness élevée pour favoriser les structures à ordre minimal.

4.2.2.3 Opérateurs de croisement et mutation: **[38]**

a) Croisement:

L'opérateur de croisement dans le cas des AGH, agit séparément sur les gènes de contrôle et les gènes de structure, et les valeurs de probabilité de croisement pc_1 et pc_2 pour la chaine de contrôle et la chaine de structure respectivement, prennent des valeurs différentes, Le type de croisement peut aussi différer pour les deux chaines, on utilise un croisement à un point de croisement pour les gênes de contrôle (codés en binaire), et un croisement à deux points de croisement pour les gênes de coefficients (codés en réels). Des résultats intéressants sont obtenus pour $pc_1 \approx pc_2/2$.

b) Mutation:

La mutation des gènes de contrôle et de structure se fait moyennant des probabilités de mutation différentes, p_{m1} et p_{m2}. Pour les gènes de contrôle, chaque bit est inversé si la probabilité p_{m1} est vérifiée, et pour les gènes de coefficients, le gène à muter sera remplacé par un autre gène pris aléatoirement dans l'espace de recherche.

4.3 Méthode de quantification proposée à base d'AG:

La méthode de quantification proposée, étant à base d'AG, garde le même processus de recherche génétique, avec tous les opérateurs connus (évaluation, sélection, ...). La particularité de notre méthode réside dans l'espace de recherche, la fonction d'évaluation ainsi que le nouveau paramètre introduit en l'occurrence, le nombre de bits (N_{bits}).

4.3.1 L'espace quantifié de la recherche:

Soit Nb le nombre de bits imposé pour coder les coefficients du filtre et q le pas de quantification qui y associé (équation (4.11)), les valeurs des coefficients sont égales a des multiples entiers de q et appartiennent donc à un espace quantifié (ou discret) appelé espace quantifié des coefficients.

$$q = 2^{-(Nb-1)} \tag{4.11}$$

L'exploration de l'espace quantifié se fait donc d'une façon quasi-aléatoire.

4.3.2 Fonction d'évaluation:

Vu le nombre restreint de valeurs dans l'espace de recherche quantifié (proportionnel à $2^{Nb} + 1$), la fonction d'évaluation est définie autrement par rapport à l'AG utilisé en travaillant en précision infinie:

Soit :

- $\varepsilon_{abs} = \frac{1}{N}\sum|H(z)_{actuel} - H(z)_{idéal}|$
- $\varepsilon_{quad} = \frac{1}{N}\sum|H(z)_{actuel} - H(z)_{idéal}|^2$

Avec : ε_{abs} : Erreur absolue moyenne

ε_{quad} : Erreur quadratique moyenne

N : Nombre d'échantillons

La fonction d'évaluation $f_{éval}$ est définie donc par:

$$f_{éval} = \begin{cases} 1/\varepsilon_{abs} & \varepsilon_{abs} > \varphi \\ 1/\varepsilon_{quad} & \varepsilon_{abs} < \varphi \end{cases}$$

Ou φ = min (δ_1, δ_2), δ_1 et δ_2 représentent l'ondulation dans la bande passante et l'atténuation dans la bande coupée du filtre respectivement.

Les solutions (filtres) instables sont pénalisées par une valeur de la fonction d'évaluation relativement faible (équation (4.6)).

4.3.3 Opérateurs génétiques

a- Sélection :

La méthode de roulette (Wheel sélection) est utilisée pour sélectionner les individus producteurs ou futurs parents, et afin d'éviter la convergence prématurée vers un individu relativement fort, on a appliqué une mise à l'échelle (scaling) à la fonction d'évaluation, comme le présente l'équation 4.12 :

$$f_{éval}^{(scal)} = a * f_{éval} + b \qquad (4.12)$$

Avec, $f_{éval}$: Fonction d'évaluation initiale ;

$f_{éval}^{(scal)}$: Fonction d'évaluation mise à l'échelle ;

a et b facteurs multiplicateurs à choisir.

b- Croisement :

On a vu ci-dessus que vu la quantification, l'espace de recherche est discret (nombre de points limités), pour cela un autre opérateur de croisement est utilisé afin faciliter la convergence du processus de recherche génétique, cet opérateur dont le principe est détaillé ci-dessous, génère de nouveaux individus avec des valeurs d'évaluation différentes et parfois élevées, à partir d'individus de la génération précédente. Par analogie à la géométrie, ce type de croisement est nommé « croisement barycentrique » **[25].**

Considérons deux chromosomes *x* et *y* appartenant à la population, les deux chromosomes contiennent des coefficients pour cellules du premier ordre $a_i^{(x)}$, $a_i^{(y)}$, et des coefficients pour cellules du second ordre $a_{j1}^{(x)}, a_{j2}^{(x)}, a_{j1}^{(y)}, et\ a_{j2}^{(y)}$, pour les polynômes du dénominateur.

Afin de satisfaire les contraintes de stabilité du filtre RII, les coefficients du terme du second ordre doivent être à l'intérieur du triangle de stabilité déterminé par :

$a_{j1} < a_{j2} + 1$, $a_{j1} > -(a_{j2} + 1)$, et $a_{j2} < 1$

Les coefficients des cellules du premier ordre doivent satisfaire donc (chapitre 1).

$$-1 < a_i < 1.$$

Le processus génétique doit assurer que les individus obtenus suite aux opérateurs de croisement et de mutation, soient toujours maintenus dans la région de stabilité. L'opérateur de croisement décrit par les équations (4.13) et (4.14) **[25],** garantie la stabilité après l'opération de croisement. Pour une paire de coefficients, si un nombre aléatoirement généré entre 0 et 1, est inferieur à la probabilité de croisement, alors un nouveau coefficient a_i est produit à partir des deux coefficients $a_i^{(x)}\ et\ a_i^{(y)}$ de la poule parent comme l'indique l'équation (4.13).

Pour les coefficients des termes du second ordre, l'équation (4.6) s'applique.

$$a_i = \alpha a_i^{(x)} + (1-\alpha) a_i^{(y)} \qquad (4.13)$$

$$a_{jk} = \alpha a_{jk}^{(x)} + (1-\alpha) a_{jk}^{(y)} \qquad (4.14)$$

La valeur du paramètre α est aléatoirement générée dans l'intervalle [0, 1], puisque le triangle de stabilité est une région convexe, la constante α dans l'intervalle [0 ,1] garantit que les nouveaux coefficients se trouveront dans la région de stabilité aussi bien après l'opération de croisement. Du point de vue géométrique, un nouveau coefficient a_i généré par l'opération de croisement se trouve sur la ligne reliant les deux coefficients $a_i^{(x)}\ et\ a_i^{(y)}$ à croisé (parents). Dans notre travail, α a été choisi en fonction de la différence de la valeur d'évaluation (fitness) afin d'accélérer la convergence, par la pondération en mettant des poids **[36]**.

$$\alpha(\Delta f) = \frac{1}{2}\Delta f + \frac{1}{2} \tag{4.15}$$

Où $\Delta f = f^{(x)} - f^{(y)}$ et $-1 < \Delta f < 1.$

c- Mutation :

Dans cette section, un autre opérateur de mutation est utilisé, il consiste à ajouté au gène à muté (codé en réel), une petite perturbation (bruit) aléatoirement généré dans l'espace de recherche suivant une distribution gaussienne, comme l'indique l'équation 4.16.

$$x^{'} = \aleph(0, \sigma) \tag{4.16}$$

On peut utiliser une probabilité de mutation p_m variable, qui diminue en fonction du nombre de générations, pour améliorer la convergence de la recherche de l'algorithme génétique **[37].**

$$\boldsymbol{p_m(k) = \frac{0.2 * e^{(k_{max}-k)/k}}{1+e^{(k_{max}-k)/k}}} \tag{4.17}$$

Où k désigne le nombre de génération actuel et k_{max} le nombre maximal de générations. La figure 4.8 montre l'évolution de la probabilité de mutation au cours des générations, au début la valeur p_m a une valeur relativement élevée (0.2) pour avoir beaucoup plus d'influence sur la recherche génétique. Cette probabilité diminue graduellement, pour tendre vers une valeur plus faible (0.05) afin de sauvegarder la performance de la recherche.

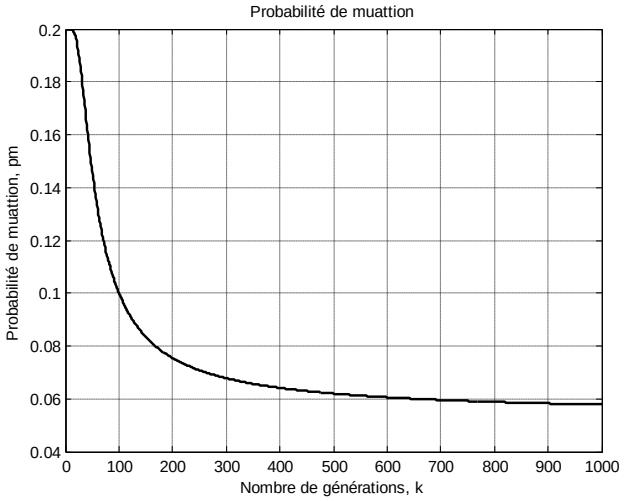


Figure 4.8 : L'évolution de la probabilité de mutation en fonction du nombre de générations

La probabilité de mutation change d'une génération à une autre pour atteindre la convergence, et enfin la technique d'élitisme assure que le meilleur individu dans chaque génération existe toujours dans la prochaine génération, pour réduire le temps de convergence.

4.4 Conclusion

Comparés aux méthodes analytiques qui utilisent le principe de la conversion de filtres analogiques à des filtres numériques, se basant essentiellement sur des transformations mathématiques, les algorithmes génétiques sont plus simples à utiliser, et plus souples pour le choix de la fonction d'évaluation (fitness), permettant ainsi d'orienter le processus de recherche en fonction du critère privilégié (stabilité, phase, gabarit, ordre, …). Cependant l'utilisation des AG, incite l'utilisateur à bien choisir le type des opérateurs génétiques (sélection, croisement, et mutation), ainsi que les valeurs des probabilités de croisement et de mutation. Le codage des chromosomes offre encore plus de souplesse à la méthode proposée pour la synthèse des filtres numériques.

CHAPITRE 5

SIMULATION ET VALIDATION DES RESULTATS

5.1 Introduction

Dans le but d'appliquer l'approche développée dans le présent travail, certains exemples de simulation ont été élaborés, et les résultats trouvés sont présentés dans ce chapitre. Pour chaque exemple présenté, ils seront inclus l'objectif, les résultats illustrés par des figures et des discussions sur leurs qualités, ainsi que des comparaisons seront établis avec les résultats obtenus par la méthode elliptique.

Le travail de simulation peut être subdivisé en deux principales parties:

Partie A :

Consiste à appliquer les algorithmes génétiques classiques et hiérarchiques pour la synthèse de filtres récursifs de différents gabarits (FPb, FPH, FPB, FCB), en précision infinie (64 bits).

Partie B :

Dans cette partie, plusieurs exemples sont présentés, montrant les résultats de la synthèse de filtres numériques RII en précision finie, en utilisant l'approche proposée. Des paramètres sont pris en considération tels que l'ordre du filtre, le nombre de bits, et la largeur de la bande passante, afin d'effectuer une comparaison des résultats avec ceux obtenus par la méthode elliptique implémentée en MATLAB, qui se base sur le principe de la transformation bilinéaire [5].

5.2 Partie A : Résultats de simulation en précision infinie

Dans cette section, on présente les résultats de la simulation, correspondants à la synthèse de filtres numériques récursifs en précision infinie (64 bits).

Pour cela, quatre filtres numériques RII (passe-bas, passe-haut, passe-bande, et coupe-bande), ont été conçus par algorithme génétique standard AGS, puis deux autres filtres (passe-bas et passe-bande) par algorithme génétique hiérarchique AGH.
Le tableau 5.1 regroupe les spécifications fixées pour les filtres FPb, FPH, FPB, FCB, en précisant que les valeurs des fréquences sont normalisées (sans unité) par rapport à la fréquence d'échantillonnage.

5.2.1 Exemple 1 : Synthèse de filtre RII par AGS en précision infinie

a- Objectif: Ce premier exemple permet d'appliquer les algorithmes génétiques standards (AGS) à la synthèse de filtres numériques de différents types (FPb, FPH, FPB, FCB), et de ce familiariser avec les différents paramètres de l'algorithme génétique, tels que le choix des opérateurs génétiques (sélection, croisement et mutation) ainsi que les probabilités de croisement et de mutation.

b- Résultats: Quatre filtres RII ont été réalisés par AGS, dont les caractéristiques sont données par le tableau 5.1, les réponses en fréquence des filtres sont représentées respectivement par les figures 5.1.a, 5.2.a, 5.3.a et 5.4.a. Les positions des pôles des filtres obtenus indiquant l'état de leurs stabilités sont illustrées par les figures respectives suivantes figure 5.1.b, 5.2.b, 5.3.b et 5.4.b. Les équations 5.1 à 5.4, donnent respectivement les expressions des fonctions de transfert des différents filtres réalisés dans l'exemple1.

Le tableau 5.2 regroupe les valeurs des différents paramètres de l'algorithme génétique AGS utilisés pour chaque cas de simulation.

Tableau 5.1: Spécifications des différents filtres conçus par AGS

Type du filtre / Caractéristiques	FPb	FPH	FPB	FCB
Fréquence de coupure normalisée ω_{p1}	0.4	0.4	0.4	0.45
Fréquence de coupure normalisée ω_{p2}	---	---	0.6	0.55
Largeur de bande de transition normalisée	0.1	0.1	0.1	0.1
Ondulation de la bande passante (dB)	0.72	0.72	0.72	0.72
Gain dans la bande coupée (dB)	35	35	35	35

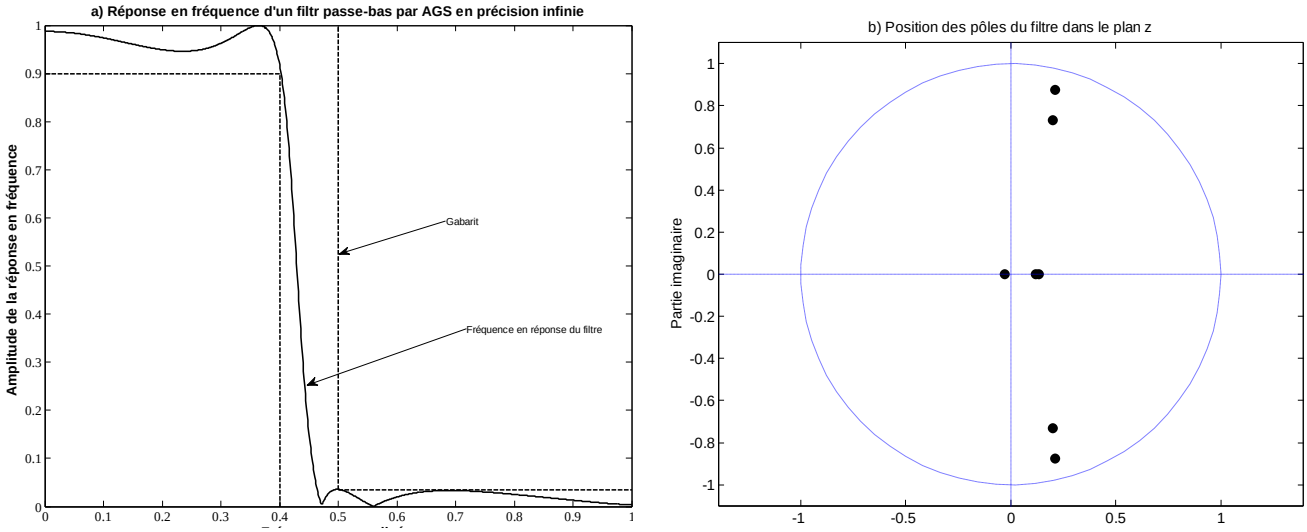


Figure 5.1 Filtre passe-bas par AGS, a) réponse en fréquence en précision infinie, b) position des pôles dans le plan z.

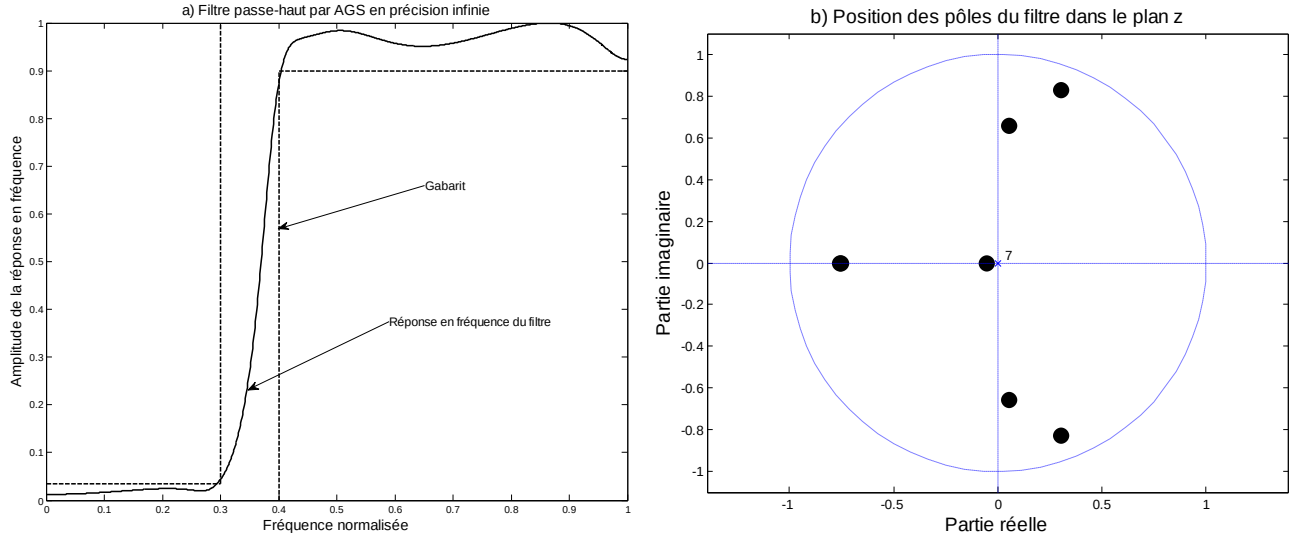


Figure 5.2 Filtre passe-haut par AGS, a) réponse en fréquence en précision infinie, b) position des pôles dans le plan z.

Les équations 5.1 à 5.2 représentent respectivement les fonctions de transfert en structure de cascade de cellules du 1er et du 2^{éme} ordre, de chacun des filtres FPb et FPH ci-dessous.

$$H_{Pb}(z) = 0.107.\left(\frac{1+0.9031z^{-1}}{1+0.2439z^{-1}}\right)\left(\frac{1+0.0306z^{-1}}{1-0.1349z^{-1}}\right)\left(\frac{1+0.2032z^{-1}}{1-0.1174z^{-1}}\right)\left(\frac{1+0.3682z^{-1}+0.9962z^{-2}}{1-0.4209z^{-1}+0.8083z^{-2}}\right)$$
$$\left(\frac{1-0.8435z^{-1}+0.9904z^{-2}}{1-0.3969z^{-1}+0.5755z^{-2}}\right) \quad (5.1)$$

$$H_{Ph}(z) = 0.2303.\left(\frac{1-0.3676z^{-1}}{1-0.4388z^{-1}}\right)\left(\frac{1-0.6793z^{-1}}{1+0.0513z^{-1}}\right)\left(\frac{1+0.7561z^{-1}}{1+0.7525z^{-1}}\right)\left(\frac{0.6007+0.0419z^{-1}+0.1566z^{-2}}{1-0.1118z^{-1}+0.4341z^{-2}}\right)$$
$$\left(\frac{-0.6495+0.1323z^{-1}+0.8558z^{-2}}{1-0.6118z^{-1}+0.7801z^{-2}}\right) \quad (5.2)$$

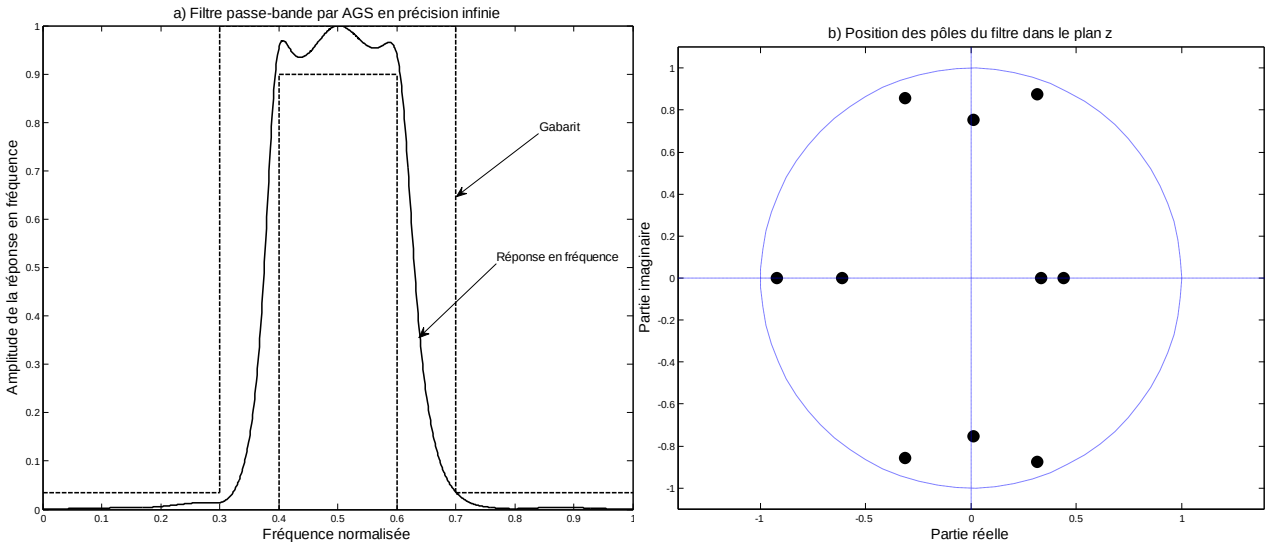


Figure 5.3 Filtre passe-bande par AGS, a) réponse en fréquence en précision infinie, b) position des pôles dans le plan z.

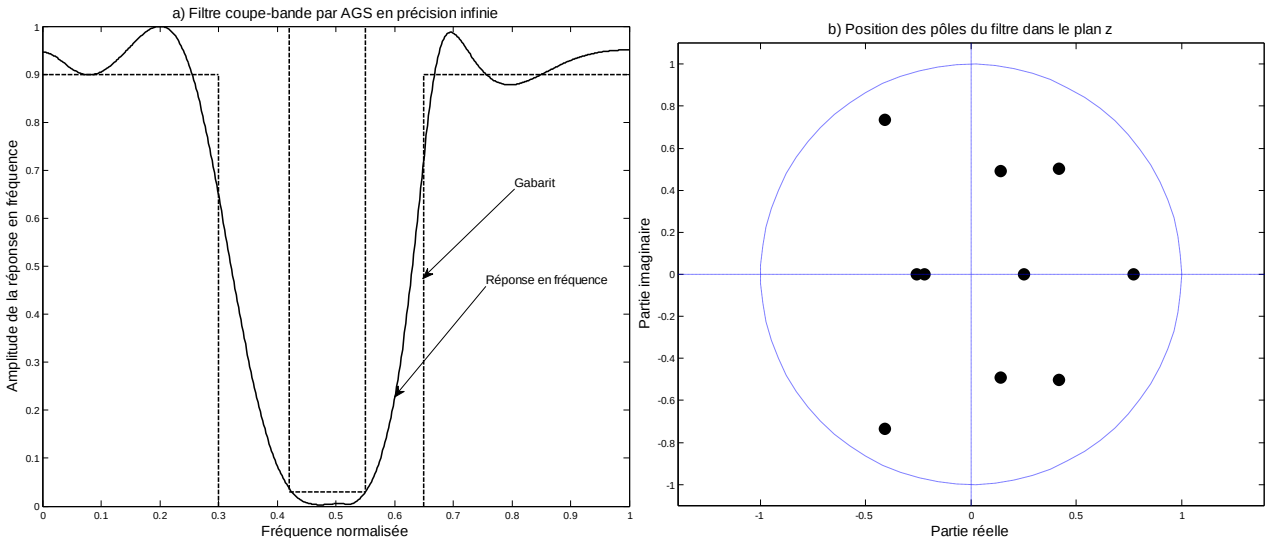


Figure 5.4 Filtre coupe-bande par AGS, a) réponse en fréquence en précision infinie, b) position des pôles dans le plan z.

Les équations 5.3 à 5.4 représentent respectivement les fonctions de transfert en structure de cascade de cellules du 1er et du 2^{éme} ordre, de chacun des filtres FPB et FCB ci-dessous.

$$H_{PB}(z) = 0.0165.\left(\frac{1-0.3327z^{-1}}{1+0.8031z^{-1}}\right)\left(\frac{1+0.9245z^{-1}}{1-0.3344z^{-1}}\right)\left(\frac{-0.0899-0.2001z^{-1}+0.8920z^{-1}}{1-0.6310z^{-1}+0.8639z^{-1}}\right)\left(\frac{0.8030-0.3906z^{-1}-0.9934z^{-1}}{1+0.1734z^{-1}-0.2694z^{-1}}\right)$$
$$\left(\frac{-0.5184-0.7333z^{-1}+0.7627z^{-1}}{1-0.0205z^{-1}+0.5684z^{-1}}\right)\left(\frac{-0.0415-0.7894z^{-1}-0.9607z^{-1}}{1+0.6267z^{-1}+0.8308z^{-1}}\right) \quad (5.3)$$

$$H_{CB}(z) = 0.2124.\left(\frac{1-0.1561z^{-1}}{1-0.2085z^{-1}}\right)\left(\frac{1-0.7707z^{-1}}{1+0.2583z^{-1}}\right)\left(\frac{-0.6467+0.9808z^{-1}-0.4445z^{-1}}{1+0.8166z^{-1}+0.7096z^{-1}}\right)\left(\frac{-0.6364-0.9187z^{-1}-0.5863z^{-1}}{1-0.2802z^{-1}+0.2610z^{-1}}\right)$$
$$\left(\frac{0.2387+0.9868z^{-1}+0.5879z^{-1}}{1-0.8359z^{-1}+0.4259z^{-1}}\right)\left(\frac{-0.4074+0.9666z^{-1}-0.5893z^{-1}}{1-0.0311z^{-1}+0.0557z^{-1}}\right) \quad (5.4)$$

c- Discussions:

Pour chaque type de filtre, on a fait une dizaine de simulations, et les moyennes de ces simulations sont illustrées par les figures 5.1.a, 5.2.a, 5.3.a, et 5.4.a. On remarque que le gabarit fixé (tableau 5.1) a été bien respecté, soit pour le niveau des ondulations autorisées dans la bande passante et la bande coupée, soit pour la largeur de la bande de transition.
Sachant qu'à la fréquence de coupure, l'amplitude de la réponse en fréquence du filtre doit être égale ou supérieure à -3dB, dans nos réalisations, cette valeur est largement dépassée, assurant une bonne caractéristique du filtre.
La stabilité étant un paramètre très important pour un filtre numérique, cette condition a été imposée dans le processus génétique utilisé, et les résultats illustrés par les figures 5.1.b, 5.2.b, 5.3.b et 5.4.b montrent clairement que les filtres obtenus sont stables.

Dans le tableau ci-dessous, on a collecté les valeurs des paramètres génétiques qui ont donné de meilleurs résultats, et une meilleure convergence.

Tableau 5.2 : Paramètres de l'algorithme génétique utilisés pour les simulations de l'exemple 1

Paramètre / Type de filtre	N_{POP}	Type de sélection	Type de croisement	Type de mutation	p_c	p_m	Gen_max
FPb	40	roulette	à un point de croisement	Changement aléatoire	0.96	0.05	1000
FPH	40	roulette	à un point de croisement	Changement aléatoire	0.96	0.05	1000
FPB	50	roulette	à deux points de croisement	Changement aléatoire	0.98	0.04	1500
FCB	60	roulette	à deux points de croisement	Changement aléatoire	0.98	0.04	2000

Npop : Nombre de chromosomes par population ;
p_c : Probabilité de croisement ;
p_m : Probabilité de mutation ;
Gen_max : Nombre maximal de génération (critère d'arrêt).

5.2.2 Exemple 2 : Synthèse de filtres RII par AGH en précision infinie

a- Objectif : En plus de son caractère génétique, l'algorithme génétique hiérarchique permet aussi d'optimiser la structure de la fonction de transfert, et par conséquent, minimiser l'ordre du filtre obtenu. Afin de comparer les résultats d'un AGH et ceux du AGS, et des filtres classiques (Butterworth, chebyshev, et elliptique), deux filtres RII ont été réalisés (passe bas et passe bande), dont les spécifications sont représentées dans le tableau 5.3, les valeurs des fréquences sont données en unité normalisée par rapport à la fréquence d'échantillonnage.

Tableau 5.3 : Spécifications des filtres conçus par AGH

Caractéristiques / Type du filtre	ω_{p1} (normalisée)	ω_{p2} (normalisée)	δ_1(d)	δ_2(dB)	Largeur de la bande de transition (normalisée)
Filtre Passe-bas	0.4	-	1	35	0.1
Filtre Passe-bande	0.4	0.6	1	35	0.1

b- Résultats : Deux filtres numériques récursifs, passe-bas et passe-bande ayant les caractéristiques du tableau 5.3, ont été réalisés par la technique à base d'algorithme génétique hiérarchique. Après plusieurs simulations, une réponse en fréquence moyenne est illustrée par la figure 5.5.

Plusieurs fonctions implémentées dans MATLAB, telles que (Buttord, Chebord, Elliord) permettent de déterminer en fonction des caractéristiques fixées, l'ordre minimum possible. Les résultats de ces fonctions seront comparés à ceux obtenus par AGH, et le tableau 5.4 illustre ceci.

Tableau 5.4 : Comparaison de l'ordre obtenu par AGH et d'autres méthodes classiques

Méthode utilisée / Type du filtre	Butterworth	Tchebychev	elliptique	AGS	AGH
FPb	26	9	6	7	4
FPB	11	6	4	7	4

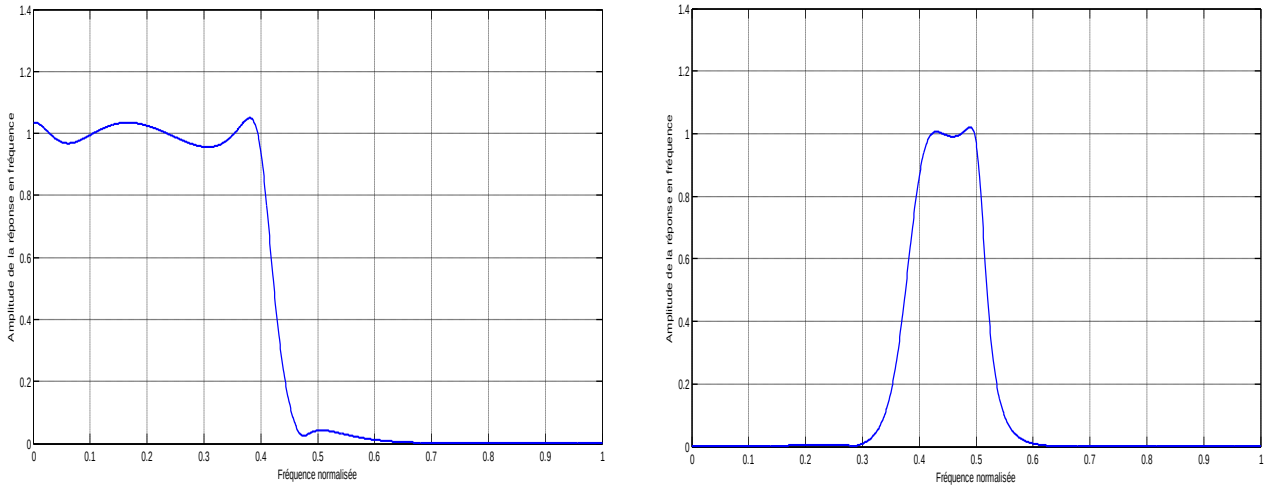


Figure 5.5 Réponse en fréquence en précision infinie par AGH, a) filtre passe-bas, b) filtre passe-bande

c- Discussions :

Le principal avantage des AGH par rapport aux AGS, consiste à l'optimisation de l'ordre du filtre, cependant le tableau 5.4 présente une comparaison entre l'ordre des filtres réalisés par AGH et ceux réalisés par AGS et par d'autres méthodes classiques implémentées dans le logiciel MATLAB (Butterworth, Tchebychev, elliptique).

Les équations 5.5 et 5.6 ci-dessous, représentent respectivement les coefficients de la fonction de transfert des filtres passe-bas et passe-bande réalisés par AGH.

$$H_{pb}(z)=0.3026*\frac{\left(1\text{-}0.8111*z^{-1}\right)}{\left(1+0.6033*z^{-1}\right)}\frac{\left(1\text{--}0.1882*z^{-1}\right)}{\left(1+0.1507*z^{-1}\right)}\frac{\left(1\text{-}0.6393*z^{-1}\text{-}0.6890*z^{-2}\right)}{\left(1+0.6040*z^{-1}\text{-}0.3886*z^{-2}\right)} \quad (5.5)$$

$$H_{pB}(z) = 0.1428\left(\frac{\text{-}0.3856\text{-}0.1407z^{-1}+0.6795z^{-2}}{1+0.6850z^{-1}+0.6534z^{-2}}\right)\left(\frac{0.4381-0.2506z^{-1}-0.6106z^{-2}}{1+0.6596\ z^{-1}-0.3507\ z^{-2}}\right) \quad (5.6)$$

5.3 Partie B : Résultats de simulation de filtres numériques en précision finie

Cette section est consacrée à l'objectif principal de notre travail, qui consiste à développer une méthode de synthèse de filtres numériques en précision finie, utilisant les algorithmes génétiques. Les coefficients de ces filtres sont représentés en un nombre fini de bits, contrairement aux résultats du logiciel MATLAB (l'outil utilisé) qui sont donnés avec

une précision très élevée (64 bits), qu'on considérera le long de notre travail comme une précision infinie.
La méthode proposée se base essentiellement sur le principe de la recherche génétique de solutions dans un domaine discret (quantifié).

Les résultats obtenus seront comparés, à ceux obtenus suite à la quantification par arrondi des coefficients de filtres obtenus par les méthodes classiques (transformation bilinéaire, …) implémentés dans MATLAB. Cette quantification sera réalisée moyennant la formule 5.7**. [12]**

$$x_q = [round(x * 2^{nb})]/2^{nb} \tag{5.7}$$

Où : *x* désigne la valeur non quantifié, x_q la valeur quantifiée et *nb* le nombre de bit.

L'algorithme proposé a été appliqué à plusieurs exemples, et les résultats obtenus sont présentés dans cette section.

5.3.1 Exemple 3 : Effets du nombre de bits

On considère dans cet exemple, les caractéristiques d'un filtre passe bande donnée par la figure 5.6. La bande passante est $\omega_p \in [0.4, 0.5]$, la bande coupée est $\omega_s \in [0, 0.3] \cup [0.6, 1]$, et la bande de transition est $\omega_t \in [0.3, 0.4] \cup [0.5, 0.6]$. Le filtre est réalisé en une structure de cascade de 2 cellules du 1^{ér} ordre, et 4 cellules du 2^{éme} ordre, correspondant à un ordre du filtre égal à 10.

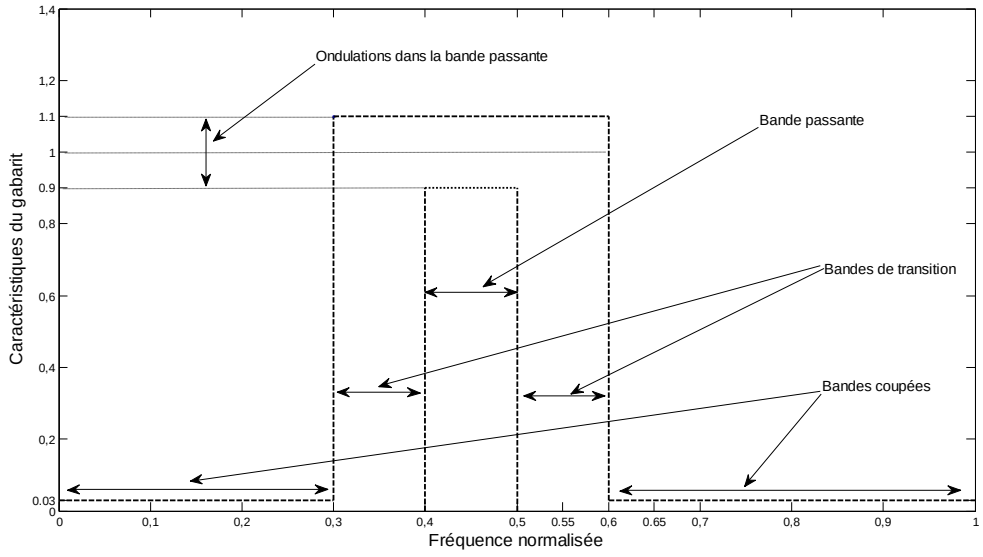


Figure 5.6 Caractéristiques du filtre de l'exemple3

a- Objectif : L'objectif de cet exemple consiste à appliquer l'algorithme proposé à la conception d'un filtre passe-bande, en précision finie, pour différentes valeurs du nombre de bits, afin de voir le comportement de notre méthode en fonction de la variation de ce paramètre (nombre de bit).

b- Résultats : On a fait la conception du filtre passe-bande de l'exemple1, pour plusieurs valeurs du nombre de bits, allant de 4bits à 64 bits, ce qui nous donne environ 15 réalisations. Uniquement les résultats correspondants au nombre de bits égal à 8, 10 et 12 sont représentés respectivement par les figures 5.7, 5.9, et 5.11, concernant les réalisations restantes, leurs résultats sont illustrés par la figure 5.13, sous forme d'un calcul d'erreur entre la réponse obtenue et la réponse voulue dans la bande passante.
Le tableau 5.5, regroupe les valeurs des paramètres génétiques utilisés dans cet exemple.

Tableau 5.5 Paramètres génétiques utilisés par la méthode proposée pour chaque cas de l'exemple 3

Paramètres / **Nb de bits**	Nombre de chromosomes par population	Probabilité de croisement	Probabilité de mutation	Nombre max de générations
8 bit	50	0.98	0.04	1500
10 bit	50	0.97	0.03	2000
12 bit	50	0.97	0.03	2000

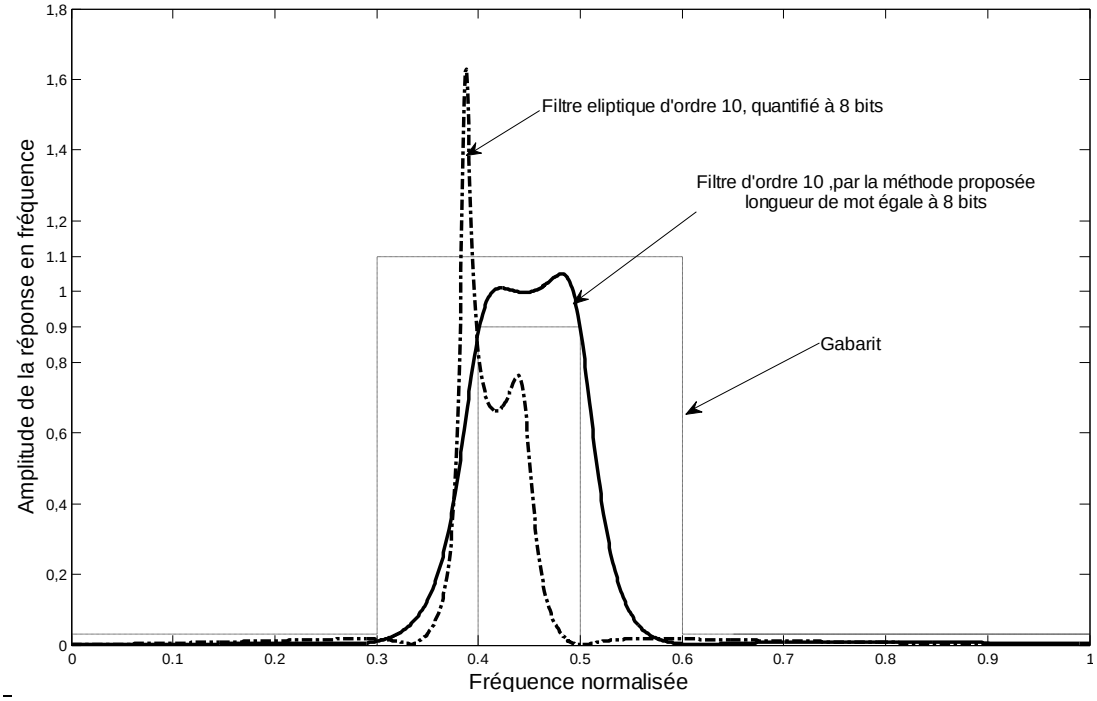


Figure 5.7 : Réponse en fréquence du filtre Passe bande, filtre elliptique quantifié à 8 bits, et Filtre en précision finie (8 bits) par la méthode proposée

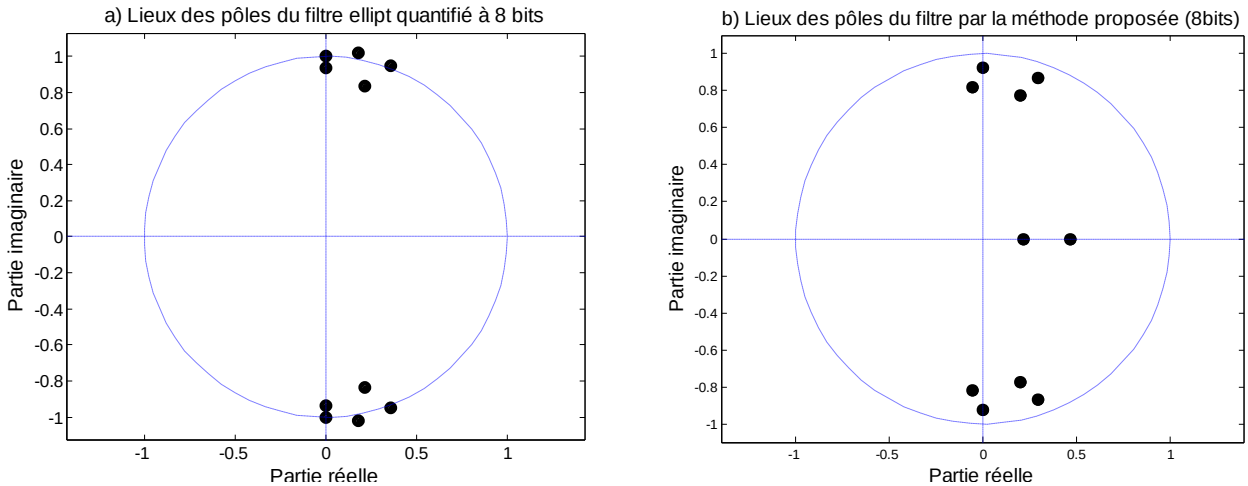


Figure 5.8: Position des pôles dans le plan *z*, a) filtre elliptique quantifié à 8 bits, b) Filtre en Précision finie (8 bits) par a méthode proposée.

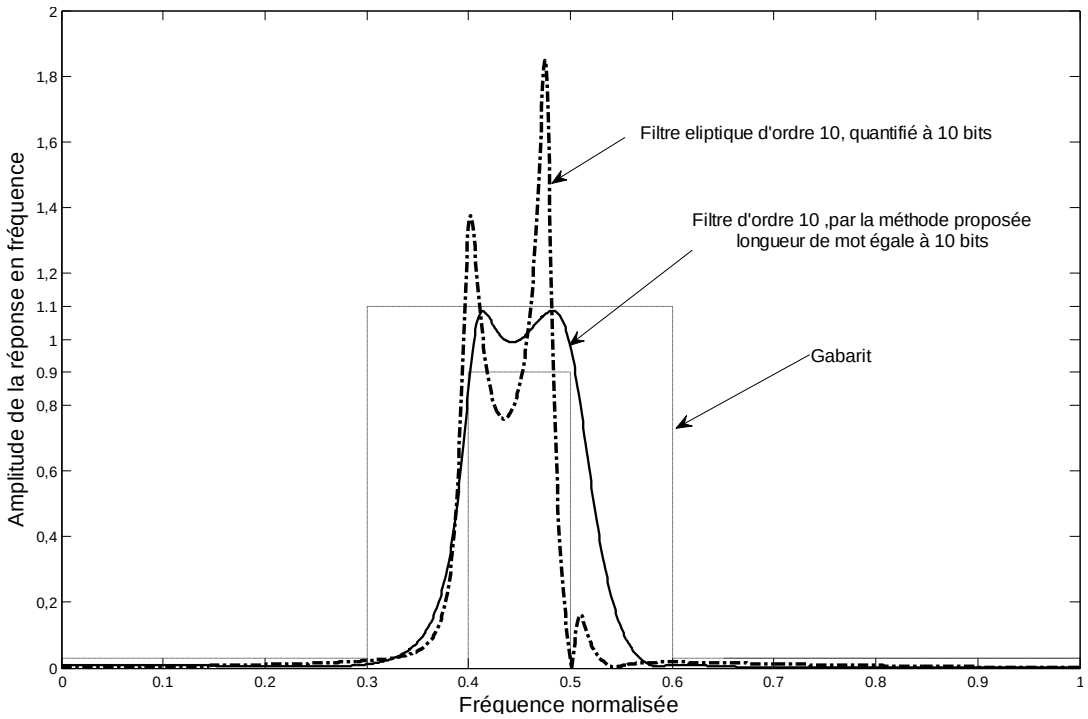


Figure 5.9 : Réponse en fréquence du filtre Passe bande, filtre elliptique quantifié à 10 bits, et Filtre en précision finie (10 bits) par la méthode proposée

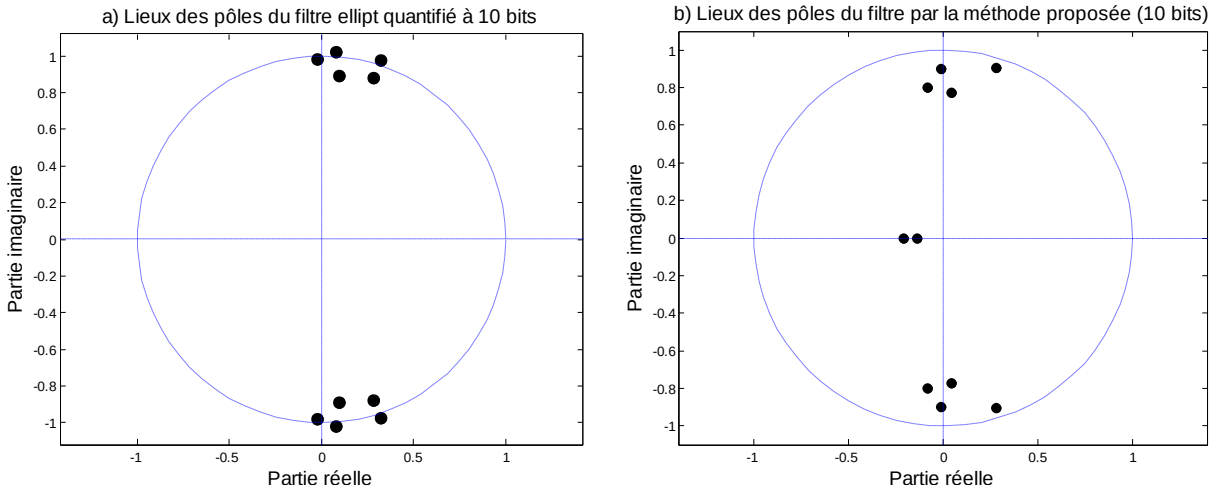


Figure 5.10: Position des pôles dans le plan *z*, a) filtre elliptique quantifié à 10 bits, b) Filtre en Précision finie (10 bits) par a méthode proposée.

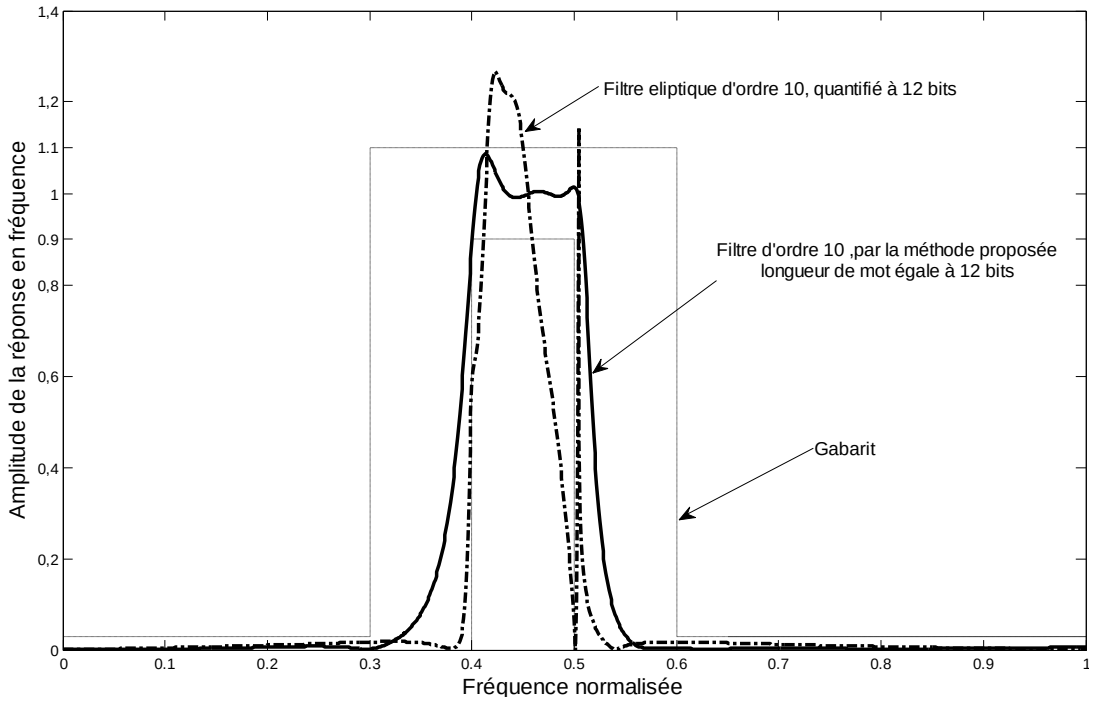


Figure 5.11 : Réponse en fréquence du filtre Passe bande, filtre elliptique quantifié à 12 bits, et Filtre en précision finie (12 bits) par la méthode proposée

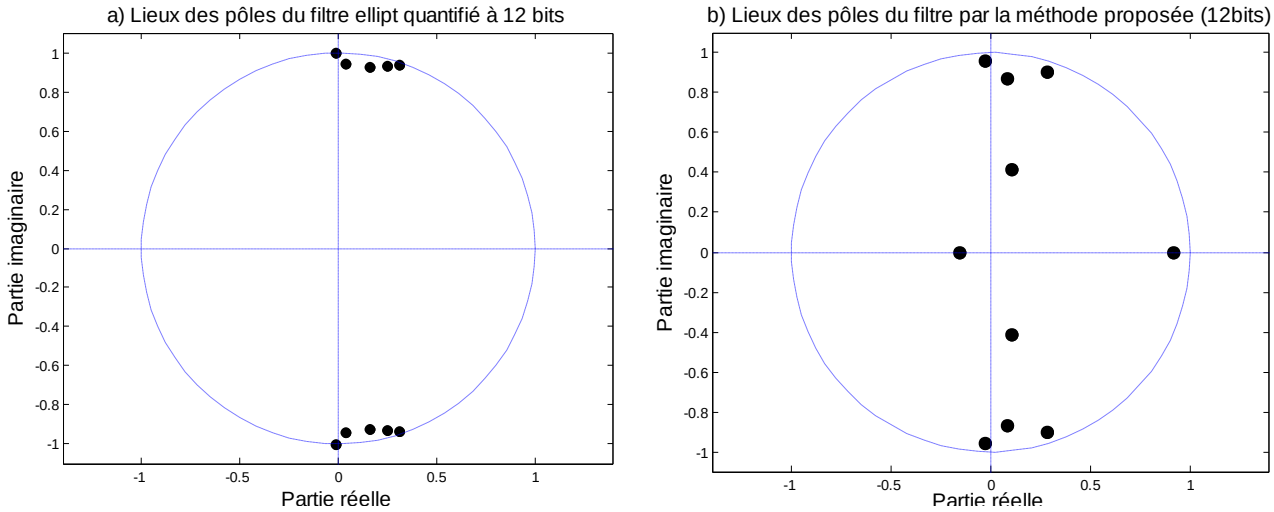


Figure 5.12: Position des pôles dans le plan *z*, a) filtre elliptique quantifié à 12 bits, b) Filtre en Précision finie (12bits) par la méthode proposée.

c- Discussions

En faisant varier le nombre de bits, on remarque d'après les figures 5.7, 5.9 et 5.11, que les résultats obtenus par la méthode proposée, répondent bien aux exigences du filtre (gabarit), par contre les filtres elliptiques quantifiés au même nombre de bits sont visiblement hors de notre gabarit.

Concernant la stabilité, la méthode proposée nous offre, d'après les figures 5.8, 5.10, et 5.12, des filtres stables et avec une marge de stabilité assez sécurisante, contrairement aux filtres obtenus par la méthode classique qui sont devenus de plus en plus instables, avec la diminution de la longueur du mot.

La figure 5.13, permet de voir l'évolution de l'erreur (dans la bande passante) en fonction de la longueur du mot binaire (nombre de bits), on remarque dans ce cas, que les filtres elliptiques nécessitent un nombre de bits assez important (> 25) pour s'approcher du gabarit voulu, la méthode proposée quant à elle donne des résultats satisfaisants pour un nombre de bits relativement faible (à partir de 6 bits).

Les équations 5.8 à 5.10 représentent les fonctions de transfert avec les coefficients réels en précision finie (8, 10 et 12 bits) des filtres passe bande obtenus par la méthode proposée.

Chacun des coefficients ci-dessous est représenté en un nombre fixe de bits (8, 10 ou 12), et peut être donc écrit sous forme d'une sommation ou soustraction de puissance de 2.

$$H_8(z) = 0.0117 * \left(\frac{1-0.8828z^{-1}}{1-0.2266z^{-1}}\right)\left(\frac{1-0.4922z^{-1}}{1-0.4609z^{-1}}\right)\left(\frac{-0.4531+0.0156z^{-1}+0.6797z^{-2}}{1-0.0625z^{-1}+0.7813z^{-2}}\right)\left(\frac{-0.3828+0.2734z^{-1}-0.7891z^{-2}}{1-0.3594z^{-1}+0.6484z^{-2}}\right)$$
$$\left(\frac{0.5234+0.9375z^{-1}-0.2109z^{-2}}{1-0.5781z^{-1}+0.8203z^{-2}}\right)\left(\frac{-0.1641-0.0469z^{-1}+0.0156z^{-2}}{1+0.1172z^{-1}+0.7266z^{-2}}\right) \quad (5.8)$$

$$H_{10}(z) = 0.0039 * \left(\frac{1+0.9199z^{-1}}{1+0.2041z^{-1}}\right)\left(\frac{1+0.7275z^{-1}}{1+0.1357z^{-1}}\right)\left(\frac{0.3184-0.8828z^{-1}+0.9902z^{-2}}{1+0.0029z^{-1}+0.7207z^{-2}}\right)\left(\frac{0.1631+0.8389z^{-1}-0.3730z^{-2}}{1+0.0537z^{-1}+0.6484z^{-2}}\right)$$
$$\left(\frac{0.0215-0.6953z^{-1}+0.9189z^{-2}}{1+0.0352z^{-1}+0.6777z^{-2}}\right)\left(\frac{0.8379+0.8223z^{-1}-0.3789z^{-2}}{1-0.5566z^{-1}+0.8848z^{-2}}\right) \quad (5.9)$$

$$H_{12}(z) = 0.0078 * \left(\frac{1-0.7363z^{-1}}{1+0.1538z^{-1}}\right)\left(\frac{1-0.5039z^{-1}}{1-0.9175z^{-1}}\right)\left(\frac{0.5190-0.7510z^{-1}-0.9775z^{-2}}{1-2085z^{-1}+0.1841z^{-2}}\right)\left(\frac{-0.2539+0.7407z^{-1}-0.7178z^{-2}}{1-0.1685z^{-1}+0.7632z^{-2}}\right)$$
$$\left(\frac{-0.5859+0.2769z^{-1}+0.6074z^{-2}}{1-0.5688z^{-1}+0.8931z^{-2}}\right)\left(\frac{-0.0024+0.3550z^{-1}-0.7583z^{-2}}{1+0.0522z^{-1}+0.9229z^{-2}}\right) \quad (5.10)$$

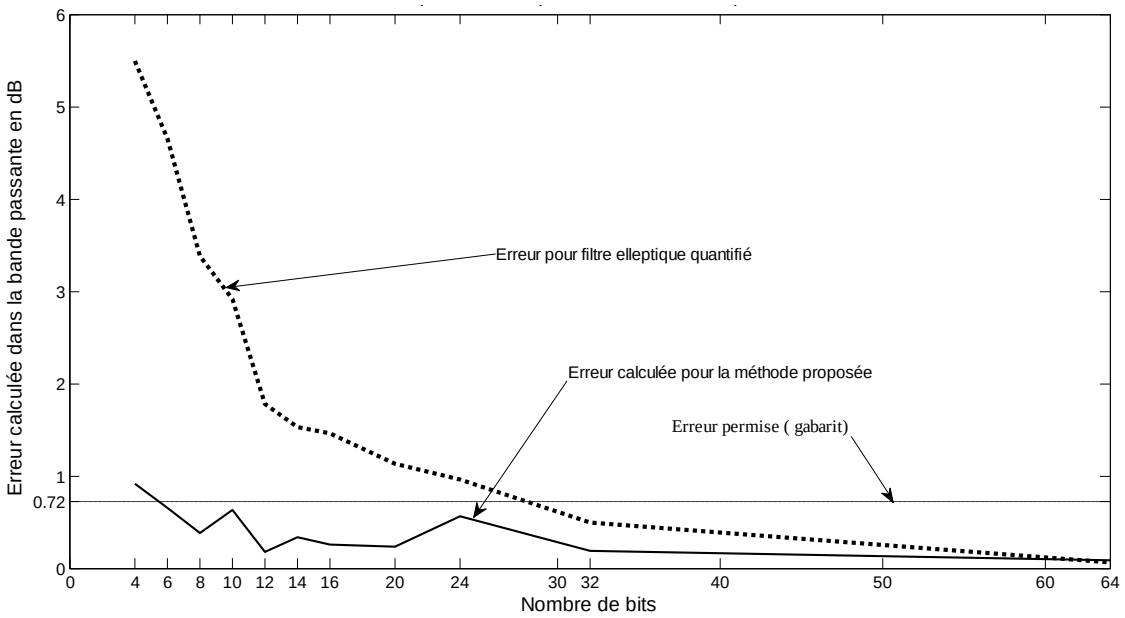


Figure 5.13: Erreur calculée dans la bande passante en fonction de la longueur du mot (nombre de bits)

5.3.2 Exemple 4 : Effets de l'ordre du filtre

Pour voir l'effet de l'ordre du filtre sur l'évolution des erreurs de quantification, on représente dans cette section les résultats de la synthèse d'un filtre passe bande, ayant les mêmes caractéristiques (gabarit) de la figure 5.6, à savoir, la bande passante est $\omega_p \in [0.4, 0.55]$, la bande coupée est $\omega_s \in [0, 0.3] \cup [0.65, 1]$, et la bande de transition est $\omega_t \in [0.3, 0.4] \cup [0.55, 0.65]$, en variant l'ordre du filtre (nombre de cellules).

a-Objectif : Il consiste à appliquer la méthode proposée à la conception d'un filtre passe-bande, en précision finie, pour des valeurs d'ordre différentes, afin de voir le comportement de notre méthode en fonction de la variation de l'ordre du filtre.

Le filtre est réalisé en une structure de cascade de m cellules du 1^{er} ordre, et n cellules du $2^{éme}$ ordre, correspondant à un ordre total du filtre égal à (2n+m).

Le choix de « n » et « m » permet donc de choisir l'ordre de notre filtre, qui varie dans cet exemple de 8 à 20.

b- Résultats : On a fait la conception du filtre passe-bande de l'exemple 2, pour plusieurs valeurs de l'ordre, allant de 8 à 20. Les résultats correspondants aux valeurs d'ordre 10, 12 et 14 sont représentés respectivement par les figures 5.14, 5.16, et 5.18, concernant les réalisations restantes, les résultats sont illustrés par la figure 5.20, sous forme d'un calcul d'erreur entre la réponse obtenue et la réponse voulue dans la bande passante.

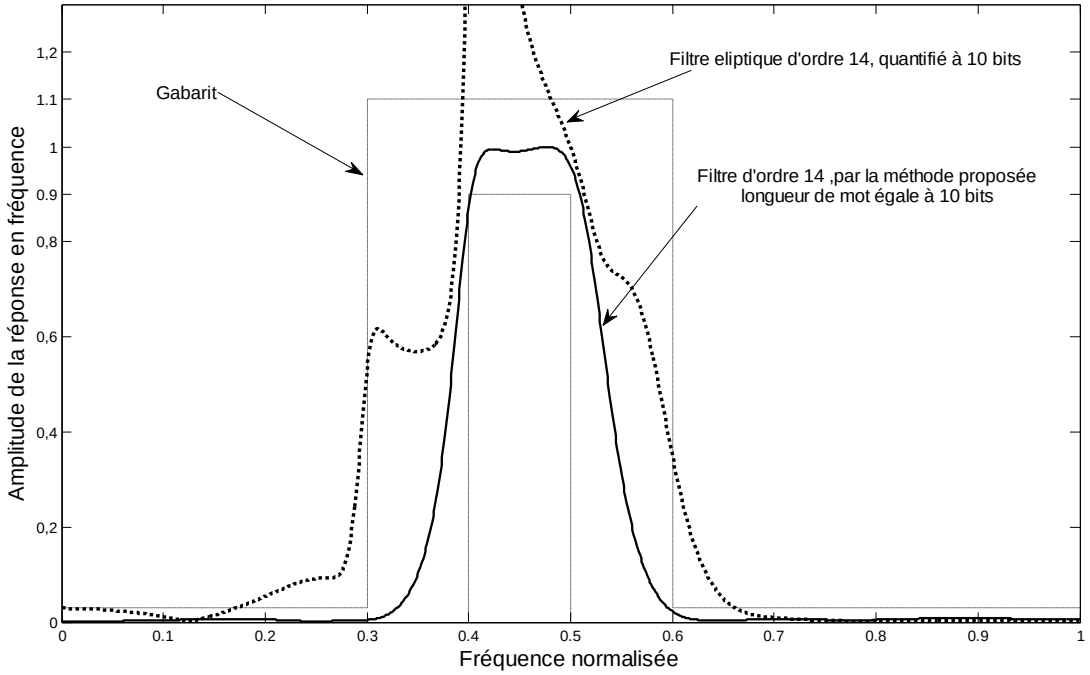


Figure 5.14 : Réponse en fréquence du filtre Passe bande d'ordre 14, a) elliptique quantifié à 10 bits, b) en précision finie (10 bits) par la méthode proposée.

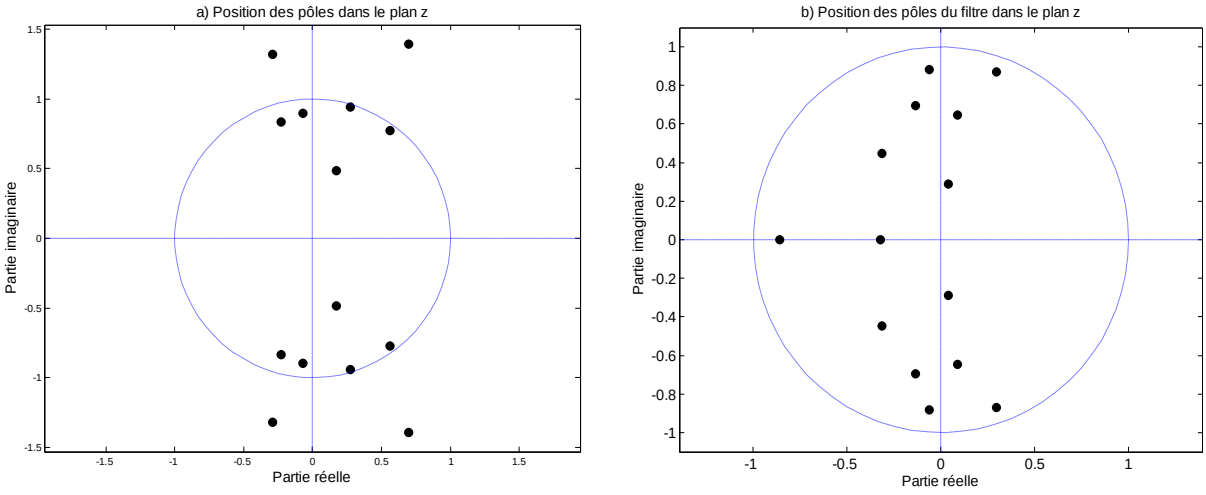


Figure 5.15: Position des pôles dans le plan z, a) filtre elliptique d'ordre 14 quantifié à 10 bits, b) Filtre en Précision finie (10bits) d'ordre 14 par la méthode proposée.

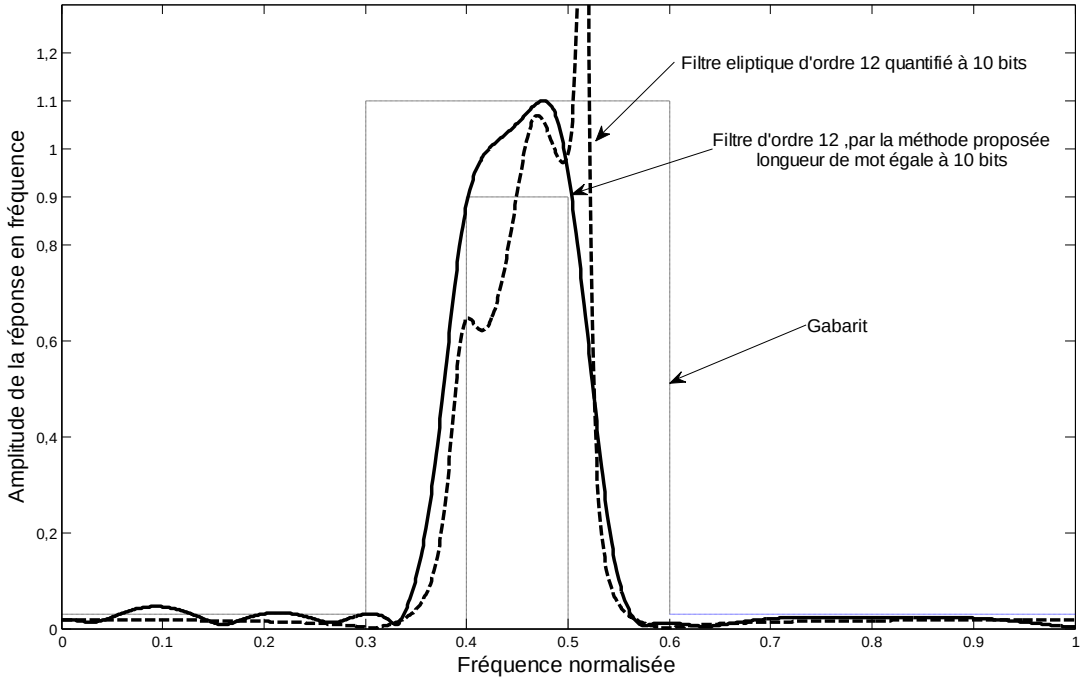


Figure 5.16 : Réponse en fréquence du filtre Passe bande d'ordre 12, a) elliptique quantifié à 10 bits, b) en précision finie (10 bits) par la méthode proposée.

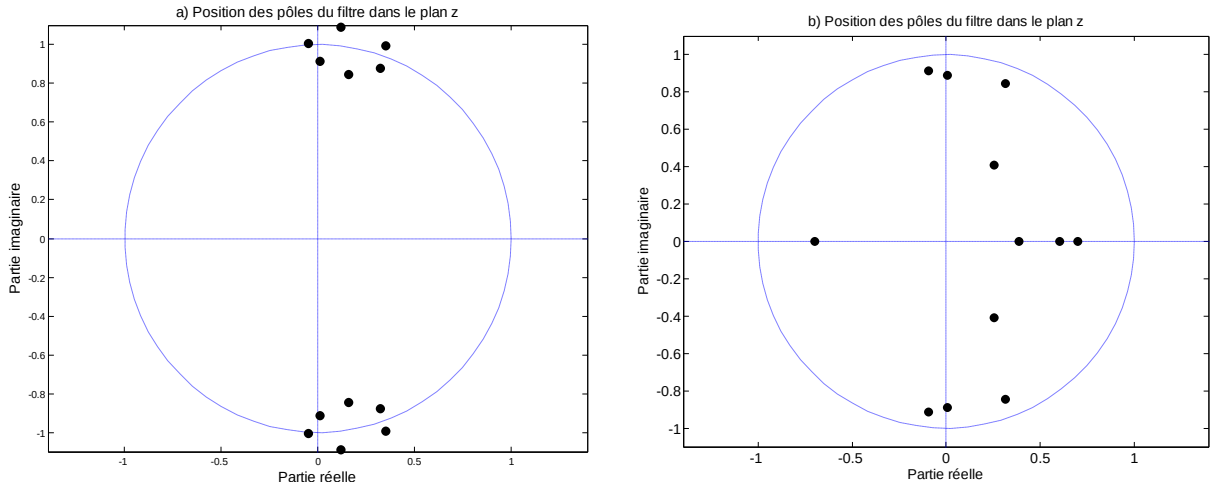


Figure 5.17: Position des pôles dans le plan *z*, a) filtre elliptique d'ordre 12 quantifié à 10 bits, b) Filtre en Précision finie (10bits) d'ordre 12 par la méthode proposée.

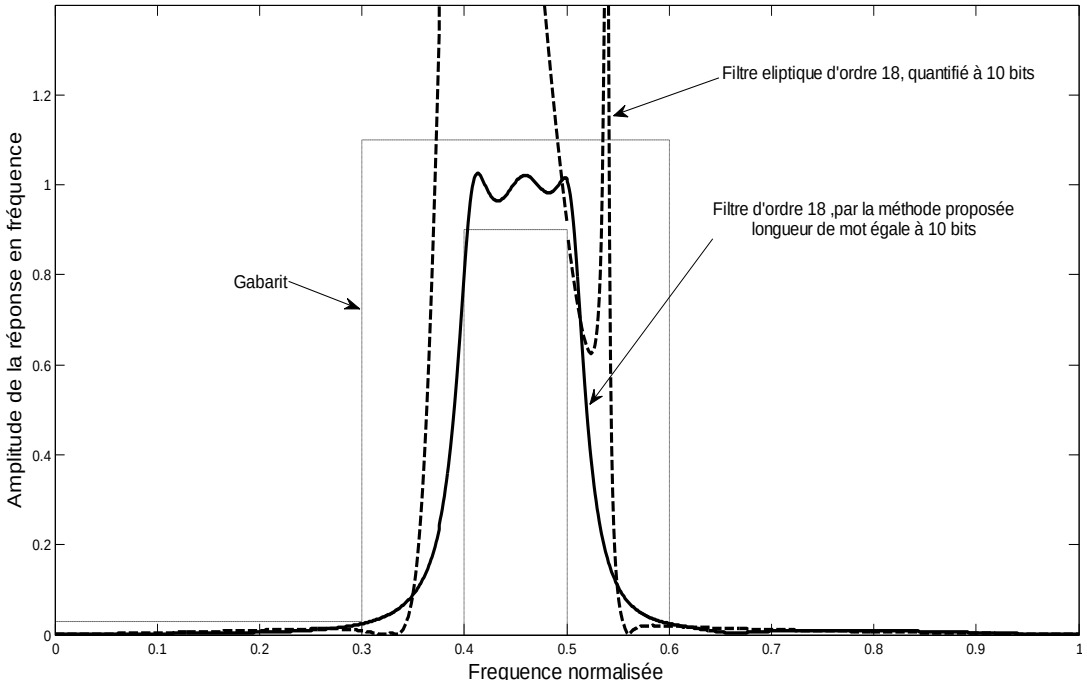


Figure 5.18 : Réponse en fréquence du filtre Passe bande d'ordre 18, a) elliptique quantifié à 10 bits, b) en précision finie (10 bits) par la méthode proposée.

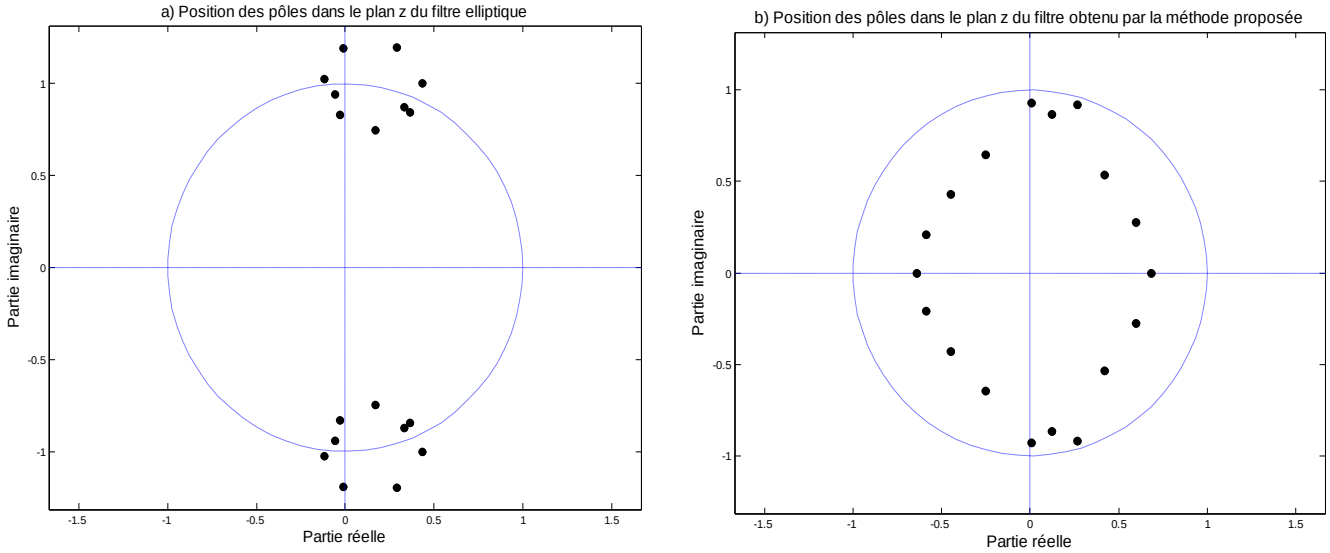


Figure 5.19: Position des pôles dans le plan *z*, a) filtre elliptique d'ordre 18 quantifié à 10 bits, b) Filtre en Précision finie (10bits) d'ordre 18 par la méthode proposée.

c- Discussions:

On faisant varier l'ordre du filtre, on remarque d'après les figures 5.14, 5.16 et 5.18, que les résultats obtenus par la méthode proposée, répondent bien aux exigences du filtre (gabarit), par contre les filtres elliptiques quantifiés au même nombre de bits sont visiblement hors de notre gabarit. Plus l'ordre du filtre augmente, plus la réponse en fréquence du filtre obtenu par la méthode elliptique est mauvaise, et ne réponde plus aux spécifications du cahier de charge, la méthode proposée quant à elle, parait presque indépendante de l'ordre du filtre, sauf que la méthode consiste à chercher un nombre plus élevé de coefficients dans un même espace de recherche, qui nécessite un nombre de générations plus élevé (tableau 5.6).

Nous savons que plus l'ordre d'un filtre augmente plus il possède de pôles, qui se rapprochent d'avantage du cercle unité, qui représente la limite de la région de stabilité. Cependant le déplacement causé par la variation des coefficients (pôles) sous l'effet de quantification, entraine une instabilité du filtre, ce qui c'est produit pour les filtres conçus par la méthode elliptique quantifiée, dont les résultats sont illustrés par les figures 5.15.a, 5.17.a, et 5.19.a. Pour la méthode proposée, le principe ne dépend pas directement de la valeur de l'ordre, il consiste seulement à choisir un ensemble de coefficients dont le nombre dépend de la valeur de l'ordre, dans un espace de recherche adéquat, tout ensemble de coefficients entrainant un filtre instable sera automatiquement refusé. Cela justifie que tous les filtres obtenus par la méthode proposée sont toujours stables comme l'indiquent les figures 5.15.b, 5.17.b, et 5.19.b.

Tableau 5.6 Nombre de générations et de coefficients, correspondants à l'ordre du filtre réalisé

Ordre du filtre		**8**	**10**	**16**	**20**	**50**	**100**	**200**	**250**	**500**
Nombre de cellules	**m**	2	2	4	2	10	20	60	50	100
	n	3	4	6	9	20	40	70	100	200
Nombre de coefficients		19	24	38	49	120	240	470	600	1200
Nombre de générations		1500	2000	200	2500	3800	4000	5500	6500	8000

m : Nombre de cellules du 1er ordre (2 coefficients par cellule);
n : Nombre de cellules du 2^{éme} ordre (5 coefficients par cellule) ;
Ordre du filtre = 2n + m ;
Nombre de coefficients = 5n+2m.

La figure 5.20, permet de voir l'évolution de l'erreur dans la bande passante en fonction de l'ordre du filtre.

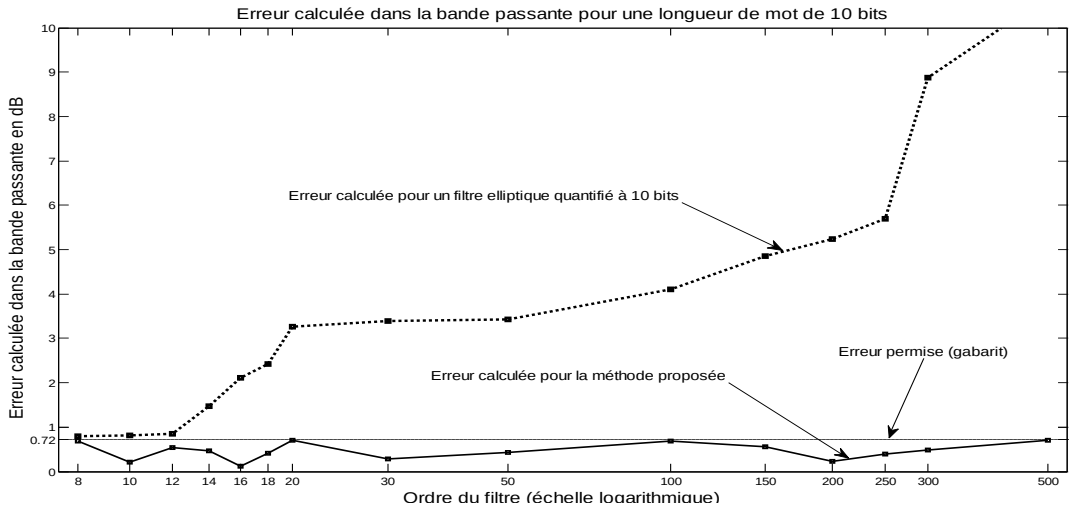


Figure 5.20: Erreur calculée dans la bande passante en fonction de l'ordre du filtre

5.3.3 Exemple 5 : Filtres à ordre élevé

a- Objectif : Dans l'exemple précédent (exemple 4), on a choisi des valeurs d'ordre < 20, l'objectif de cet exemple, consiste à concevoir un filtre à ordre très élevé afin de voir les limites de la méthode proposée. La longueur du mot (nombre de bits) choisi pour ce travail est de 10 bits.

b- Résultats : On a fait la conception du filtre passe-bande de l'exemple 4, pour plusieurs valeurs de l'ordre, allant de 30 à 500. Les résultats correspondants aux valeurs d'ordre 62, 100 et 150 sont représentés par la figure 5.21, concernant les réalisations restantes, les résultats sont illustrés par la figure 5.20, sous forme d'un calcul d'erreur entre la réponse obtenue et la réponse voulue.

c- Discussions:

Les résultats de ce deuxième exemple, montrent clairement que la méthode proposée et le processus génétique en général est une méthode qui ne dépend presque pas de l'ordre du filtre. Pour notre méthode, le principe consiste à choisir un nombre de coefficients fonction de l'ordre du filtre.

En augmentant le nombre des cellules, et par conséquent l'ordre des filtres, on remarque d'après la figure 5.21, que les résultats sont toujours satisfaisants, sauf que cela nécessite plus de générations pour atteindre l'optimum voulu, comme l'indique le tableau 5.6.

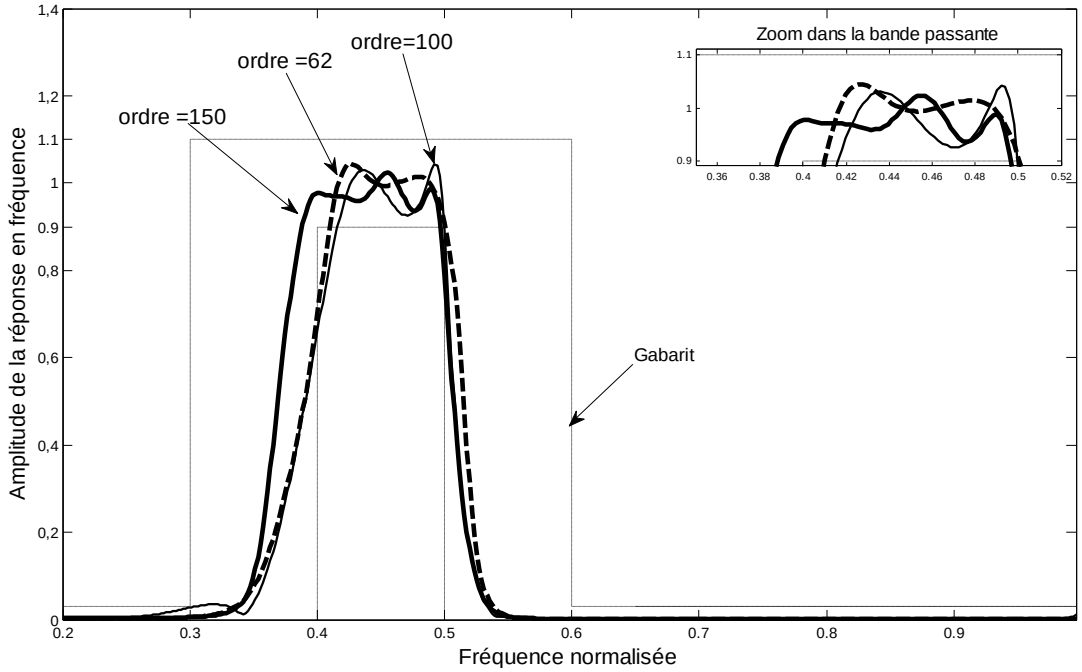


Figure 5.21 Réponses en fréquence des filtres passe-bande d'ordres 62, 100 et 150 obtenus par la méthode proposée à 10 bits.

5.3.4 Exemple 6 : Effets de la largeur de la bande passante du filtre

a- Objectif : Dans cet exemple, on s'intéressera à la largeur de la bande passante du filtre, qui représente elle aussi un paramètre important, en particulier dans le domaine de quantification, ou les effets de la précision finie sont de plus en plus apparents pour les faibles largeurs de bande.

Tableau 5.7 Caractéristiques des filtres de l'exemple 6

Paramètres / Filtre	Ordre	Nombre de bits	Fréquence de coupure normalisée	Largeur de bande normalisée
Filtre 1	12	10	$\omega_p \in [0.4, 0.41]$	0.01
Filtre2	12	10	$\omega_p \in [0.5, 0.505]$	0.005

b- Résultats : Le tableau 5.7 représente les caractéristiques des deux(02) filtres passe-bande, ayant un ordre égal à 12, et la longueur du mot binaire (nombre de bits) est fixée à 10 bits. Les résultats obtenus par la méthode proposée et ceux obtenus par la méthode elliptique, sont illustrés par les figures 5.22 et 5.24. Ainsi que la position des pôles indiquant l'état de stabilité de ces filtres qui est représenté par les figures 5.23 et 5.25.

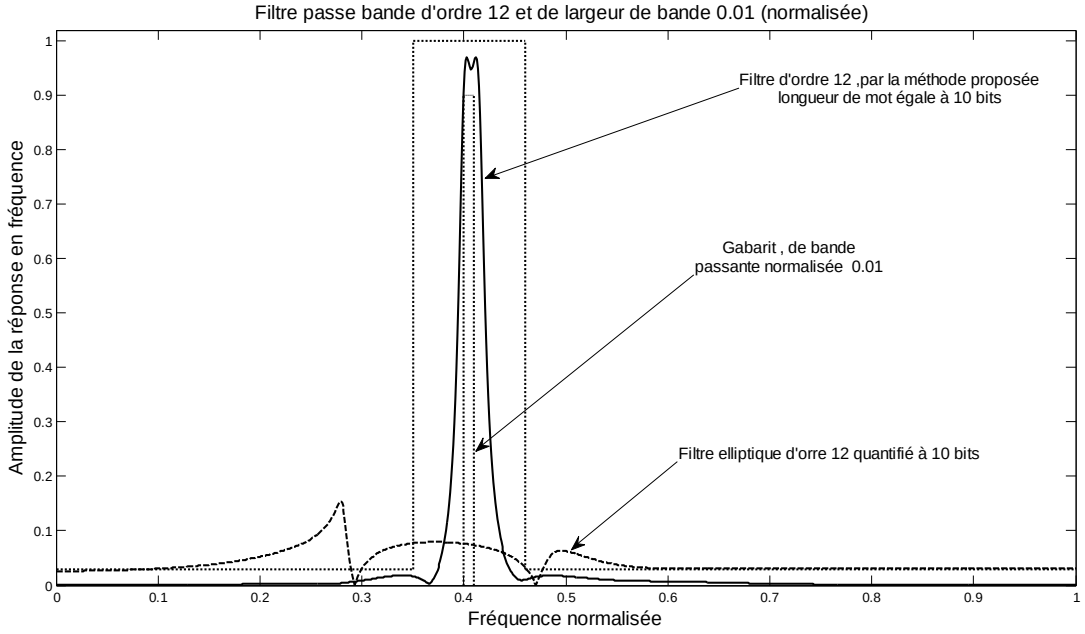


Figure 5.22 : Réponse en fréquence du filtre Passe bande d'ordre 12, a) elliptique quantifié à 10 bits, b) en précision finie (10 bits) par la méthode proposée.

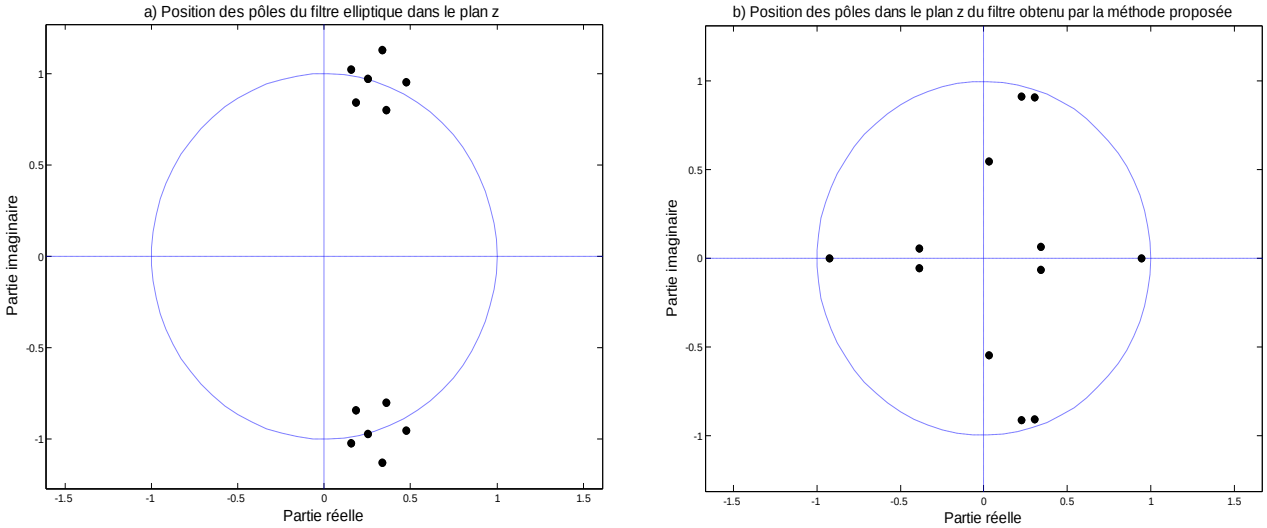


Figure 5.23: Position des pôles dans le plan *z*, a) filtre elliptique de BP= 0.01 quantifié à 10 bits, b) Filtre en Précision finie (10bits) de BP = 0.01 par la méthode proposée.

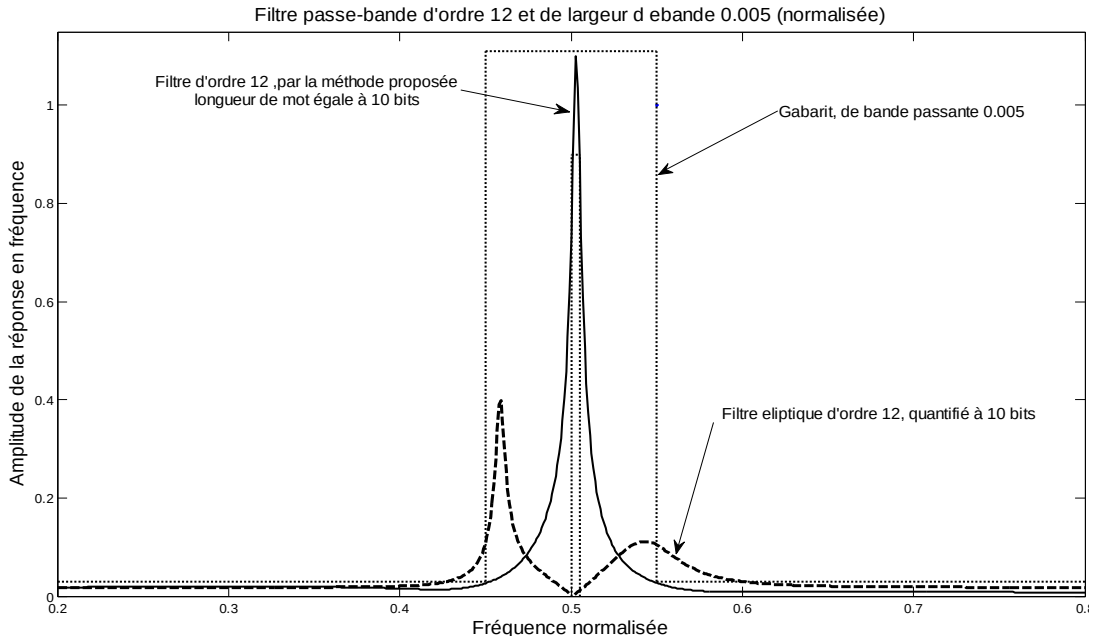


Figure 5.24 : Réponse en fréquence du filtre Passe bande d'ordre 12, bande passante normalisée 0.05 a) elliptique quantifié à 12 bits, b) en précision finie (10 bits) par la méthode proposée.

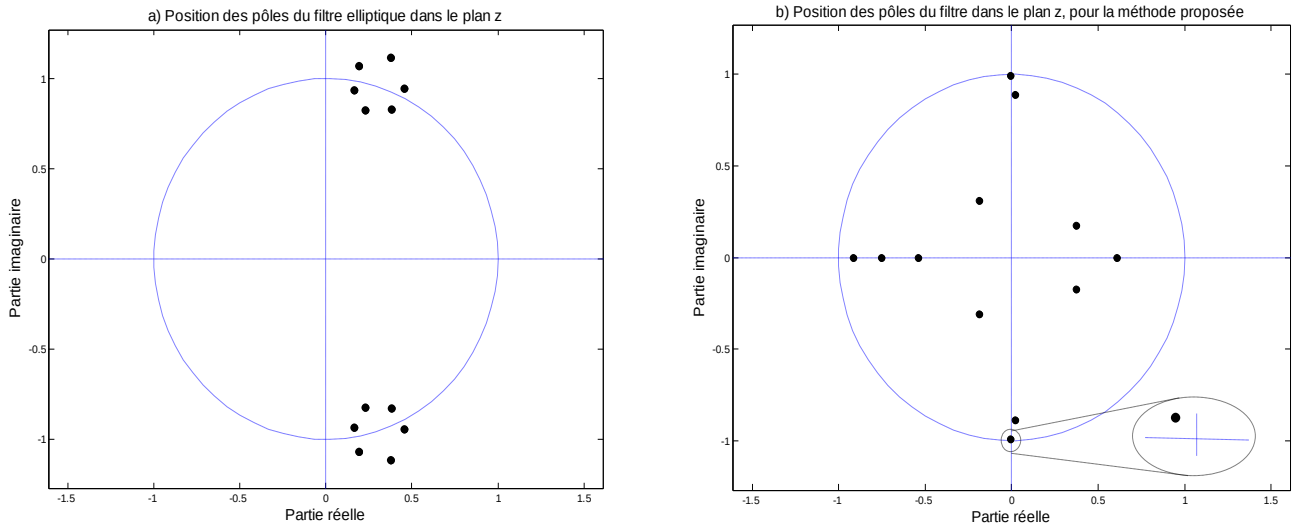


Figure 5.25: Position des pôles dans le plan *z*, a) filtre elliptique de BP= 0.005 quantifié à 10 bits, b) Filtre en Précision finie (10bits) de BP = 0.005 par la méthode proposée.

c- Discussions:

La largeur de la bande passante du filtre, favorise l'apparition des effets de quantification pour le filtre elliptique, surtout pour la stabilité. La méthode proposée, maintient la stabilité du filtre, ainsi que les exigences du gabarit.

5.4 Conclusion

Le long de ce chapitre, nous avons présenté les résultats de plusieurs simulations, de synthèse de filtres récursifs en précision finie, par la méthode proposée à base d'algorithmes génétiques. Trois principaux paramètres ont été pris en considération pour la comparaison avec les résultats donnés par les filtres elliptiques utilisant le principe de la transformation bilinéaire, à savoir, la longueur du mot binaire défini par le nombre de bit, l'ordre du filtre et enfin la largeur de la bande passante du filtre.

L'approche proposée a permis d'obtenir des filtres numériques récursifs, dont les réponses en fréquence répondent bien aux spécifications fixées par le cahier de charge du filtre, ceci pour différents paramètres cités plus haut.

Pour les longueurs de mot (supérieur à 6 bits), les filtres présentent toujours des pôles à l'intérieur du cercle unité synonyme de stabilité. Les coefficients des fonctions de transfert sont donnés donc en précision fini, c.-à-d. codés en un nombre fini de bits, choisi par l'utilisateur de la méthode, en fonction de la cible d'implémentation du filtre (DSP, FPGA, …).

Ces mêmes coefficients peuvent être représentés en une sommation de puissance de 2, ce qui réduit les opérations de multiplications à des simples décalages au niveau registre, ceci permettra de s'en passer des multiplicateurs trop gourmands en surface et en consommation.

CONCLUSION

L'objectif majeur du travail effectué dans le cadre de ce mémoire, a été d'explorer la nature flexible des algorithmes génétiques, et leurs possibilités à obtenir plusieurs solutions prometteuses dans les problèmes d'optimisation multiobjectifs liés à la synthèse de filtres numériques, dont les coefficients sont représentés en précision finie.

Dans le chapitre 5, partie A, une première approche pour la conception de filtres numériques, à base d'algorithmes génétiques a été développée. Dans cette approche les coefficients de filtres numériques RII de différents gabarits (passe-bas, passe-haut, passe-bande et coupe-bande) sont déterminés, en précision infinie, c-à-d représentés en 64 bits. Les résultats obtenus pour les 04 types de gabarits ont bien satisfait les spécifications fixées, tels que l'amplitude des ondulations dans la bande passante, le gain dans la bande coupée et la largeur de la bande de transition. La stabilité constituant un paramètre important pour le filtrage numérique, est toujours vérifiée pour les filtres obtenus dans les simulations effectuées.

La seconde approche dans cette partie, consistait à utiliser un codage des chromosomes en structure hiérarchique, cette version d'algorithmes génétiques a été exploitée pour créer une population de chromosomes (vecteurs de coefficients) de longueur variable. Les solutions candidates ont été représentées sous forme de chromosomes à deux niveaux, à savoir niveau de gènes de contrôle (codage binaire, 1 ou 0) et niveau de gènes de coefficients (réels). La longueur ou l'ordre du filtre, est déterminé par le nombre de gènes de contrôle non nuls, présents dans le chromosome solution, qui dictait l'état d'activation des gènes de coefficients correspondants. Les résultats des simulations de cette approche montrent que les filtres obtenus par AGH, satisfont les spécifications exigées, avec un ordre minimal et relativement plus faible par rapport à l'ordre donné par les méthodes implémenté dans MATLAB telles que Butterworth, Tchebythev, et elliptique.

Les développements récents dans les composants logiques programmables (PLD) et FPGA, ainsi que le principe de programmation dans le circuit de ces composants permettent des solutions à faible coût offrant d'avantage de vitesse et moins de consommation. Mais à cause des ressources limitées disponibles dans les PLDs et les FPGAs, le filtre numérique doit être implémenté en cellules de faible ordre (1^{er} et $2^{éme}$

ordre), d'une part et leurs coefficients doivent être représentés en un nombre limité et adéquat de bits d'autre part. Pour faciliter donc l'utilisation de composants programmables tels que PLDs et FPGAs, L'approche de la partie B chapitre 5, a permis de faire la synthèse de filtres numériques en structure de cascade de cellules du 1^{er} ordre et cellules du $2^{éme}$ ordre, a coefficients réels représentés en un nombre fixe de bits fixé par l'utilisateur.

Les résultats obtenus montrent l'efficacité de cette approche à base d'algorithmes génétiques, à faire la synthèse de filtres numériques RII en précision finie, sans se soucier des effets de la longueur du mot finie (nombre de bits limité), les résultats sont très intéressants du point de vue réponse en fréquence, et même la stabilité des solutions (filtres) est assurée.

Pour cela une comparaison a été faite avec la méthode elliptique implémentée dans MATLAB, en variant l'ordre du filtre (de 8 à 500), on a remarqué que plus on augmente l'ordre du filtre, plus les résultats de la méthode elliptique quantifié sont mauvaises, contrairement à l'approche proposée, dont les résultats sont presque indépendants de la longueur du filtre, sauf que la synthèse prend plus de temps (nombre de générations).

On a eu aussi de bons résultats avec un nombre de bits faible (jusqu'à 6 bits), et des largeurs de bande passante assez réduite. Les filtres obtenus par cette approche sont toujours stables quelque soit, l'ordre, le nombre de bits et même la largeur de la bande passante du filtre.

Une autre caractéristiques des filtres obtenus par l'approche proposée, se rapporte à leurs coefficients qui étant codés sur un nombre fixe de bits, peuvent être écrit sous la forme d'une sommation de puissance de 2, ceci permet de faire l'implémentation de ces filtres sans utilisation de multiplicateurs (multiplierless filters), qui sont trop gourmand en surface et en consommation dans les circuits PLDs et les FPGAs.

Notre effort à la synthèse de filtres numériques en précision finie en utilisant les algorithmes génétiques, a prouvé que cette méthodologie est très intéressante, dans le sens où on a pu obtenir des conceptions meilleures du point de vue engineering.

Les efforts dans ce domaine doivent être poursuivis pour développer de nouvelles méthodes de conception à base d'AG, pour cela certaines idées concernant des applications plus ciblées citées ci-dessous peuvent faire l'objet de travaux futurs.

Le fait de travailler en « offline » (temps différé), le paramètre temps a été plus ou moins ignoré dans l'aspect général de notre approche, pour cela une autre approche « hybride » pour la conception de filtres numériques à précision finie utilisant les AGs avec d'autres méthodes d'optimisation analytique telle que les techniques à gradient (Quasi Newton) pourra être développer [2]. Cette future approche combinera la convergence rapide et la précision de la technique à gradient avec la flexibilité et diversité des AGs. L'AG est utilisé pour la recherche globale, et la méthode à gradient est utilisé pour exploité l'information locale.

Une autre hybridation de l'AG peut être envisagée avec une autre méthode d'optimisation métaheuristique telle que la logique floue, cette approche hybride a donné déjà de bons résultats [39], et pourra bien être appliquée à la conception de filtres numériques en précision finie.

Une fois le temps d'exécution de l'approche à base de AG est contrôlé et minimisé, on pourra penser à implémenté la technique en hardware pour travailler en temps réel, nécessaire pour des applications tel que le filtrage adaptatif.

References

1 Thierry Dutoit, "Introduction à la synthèse de filters actifs", Notes de cours, Première édition Copyright 2000, Faculté polytechnique de Mons.

2 Saeed V.Vaseghi, " Advanced Digital Signal Processing, and Noise Reduction", Copyright 2006, John Wiley & Sons Ltd, The Atrium, Southern Gate, England, third edition.

3 Gabriele D'Antona, Alessandro Ferrero, " Digital Signal Processing for Measurement Systems, Theory and Applications", Copyright 2006 Springer Science+busness Media, Inc.

4 Nicolas Hervé, "Contributions à la synthèse d'architecture virgule fixe à largeurs multiples", thèse doctorat, Université de Rennes 1, mars 2007.

5 A. Antoniou. "Digital Filters: Analysis, Design, and Applications". McGraw-Hill Companies; 2 Sub edition (January 1, 1993).

6 D. Menrad, "Méthodologies de conversion automatique en virgule fixe pour les applications de traitement du signal", Ecole thématique ARCHI 03, Roscoff 31Mars–4 Avril.

7 Emmanual.C.Ifeachor, Barrie W Jervis, "Digital Signal Processing, A practical Approach" Prentice Hall, second edition 2002.

8 Lynn P.A. and Fuerst W. (1989), "Introductory Digital Signal Processing with Computer Applications" Wiley. John Wiley & Sons Ltd; edition: New edition of 2 Revised editions (29 avril 1998).

9 John G. Proakis and Dimitris G. Manolakis, "Digital Signal Processing", Prentice Hall; edition**:** United States edition of 4th revised ed (27 Avril 2006).

10 Richard G. Lyons, "Understanding digital signal processing", Prentice Hall PTR, eighth printing, April 2001.

11 Monson H. Hayes, "Schaum's outline of theory and problems of digital signal processing", Schaum's outline series, Mc Graw-Hill © 1999.

12 S.E Hadri, " Contribution à la synthèse de structures optimales pour la réalisation de filtres et de régulateurs en précision finie", Thèse Doctorat Institut National polytechnique de LORRAINE, octobre 1996.

13 Brian D.Hann, Daniel T. Valentine, "Essential MATLAB® for Engineers and Scientists ", Liance House, Jordan Hill, Oxford, Third edition 2007.

14 Leland B. Jackson, "Digital Filters and signal processing", Kluwer Academic; 3rd edition (September 30, 1995).

15 Richard G. Lyons, "Understanding Digital Signal Processing", Prentice Hall, edition: 2nd revised edition (7 avril 2004).

16 Vincent P. Heuring, Harry F. Jordan "Computer Systems Design and Architecture" Prentice Hall, 1st edition (November 5, 1996).

17 Constantine H. Houpis, Gary B. Lamont, "Digital Control Systems, Theory, Hardware, Software" Mc Graw-Hill, Inc second edition, 1992.

18 H. MEKKI, A.MELLIT, H. SALHI, K. BELHOUT," FPGA implementation of MLP neural network; application for modelling of PV panel", MCSA, USTO, 28-29-30 /04/ 2008.

19 J-M. Tourreilles, C Nouet, E. Martin, "Méthode d'implémentation d'algorithmes de traitement du signal en précision finie" Seiziéme colloque GRETSI – 15 – 19 septembre 1997 – Grenoble.

20 Romuald Rocher, Daniel Menard, Nicolas Herve, and Olivier Sentieys, "Fixed-point configurable hardware components", Hindawi publishing corporation, EURASIP, Journal on Embedded Systems, Volume 2006, Article ID 23197, pages 1-13.

21 T. Salim, J. Whittington, J. Devlin, "Quantization Effects in Digital Upconversion and Digital Beam Forming for TIGER Radar System" Hobart, Australia: National Committee for Radio Science and URSI, 2004.

22 Goldberg, D.E., "Algorithmes Génétiques "Edition Addison Wesley, France, (1994).

23 Man K.F, Tang K.S and Kwong S, "Genetic Algorithms, Concepts and Designs", Edition Springer-Verlag , (2001) .

24 Goldberg, D.E., "Genetic Algorithms in Search, Optimisation and Machine Learning ", Addison-Wesley Reading, M.A., (1989).

25 Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolutionary Programs", Springer, 3ème Edition. 1996.

26 CL Bridges and D.E Goldberg. "An analysis of multipoint crossover" In Proceedings of the foundation of Genetic Algorithm FOGA, 1991.

27 M. Nilsson, M. Dahl, Clacsson, " Digital filter design of IIR filters using real valued genetic algorithm", Proceedings of the 2nd WSEAS International Conference on Electronics, Control and Signal Processing, Singapore, 2003.

28 Arabas, J., Michalewicz Z., Mulawka J., "A Genetic Algorithm With Varying Population Size GAVaPS ", Proc of First IEEE, Conference on Evolutionary Computation, (1994), 73-78.

29 De Jong A.," An analysis of the behaviour of a class of genetic adaptive systems" PhD. Dissertation 76-9381, University of Michigan, (1975).

30 Schaffer J.D, Caruna R.A, Eshelman L.J. et Das R, "A study of control parameters online performance of genetic algorithms for function optimization" , in Proceedings of the third Int. Conf. on Genetic Algorithms, (1989), 51–60.

31 Hesser J. et Männer R., "Towards an optimal mutation probability in genetic algorithms",Lecture Notes In Computer Science; Vol. 496 , Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, p 23-32, 1990.

32 Edwin D. de Jong, Dirk Thierens, Richard A. Watson, " Hierarchical Genetic Algorithms", Lecture Notes in computer science, Springer Berlin\ Heidelberg, ISSN 0302 – 9743, December 2004.

33 Wilson, P.B and Macleod, "Low implementation cost IIR digital filter design using genetic algorithms", Proceedings of workshop on Natural algorithms in signal processing. Vol.1, pp. 4/1-4/8, Chelmsford, UK, Nov. 1993.

34 Nuhan Karaboga, Bahadir Cetinkaya, "Design of minimum phase digital IIR filters by using genetic algorithm", Proceeding of the 6th Nordic signal processing symposium, NORSIG 2004, June 9-11, 2004, Espoo, Filand.

35 Jung – doo Jang, Seong G. Kong, "Design of optimal digital IIR filters using the genetic algorithm" International Journal of Fuzzy Logic and Intelligent Systems Vol.2 No.2, 2002. 6 pp. 115~121 (7 pages).

36 K. S. Tang, K. F Man, S Kwong, Q. HE, "Genetic Algorithms and their Applications", IEEE Signal processing magazine, November 1966.

37 Kit-sang, Kim-fung, Sam Kwong, and Zhi-feng Liu, "Design and Optimization of IIR Filter Structure Using Hierarchical Genetic Algorithms", IEEE Transaction on Industrial Electronics, Vol. 45, N°.3, June 1998.

38 Gurvinder S. Baicher, "Optimization of Finite word length Coefficient IIR Digital filters through genetic algorithm – A comparative study", L. Jiao et al (Eds): ICNC 2006, Part II, lncs 4222, pp. 641 – 650, 2006 © Springer – Verlag Berlin Heidelberg 2006.

39 Abdelhalim Benouared, "Utilisation d'un controleur hybride Flou-HGA, pour le controle d'un réacteur chimique du type C.S.T.R", mémoire de Magister, Université de Blida, Algérie, juillet 2007.

APPENDICE A

LISTE DES SYMBOLES ET DES ABREVIATIONS

µcontrôleur	:	Micro contrôleur
Δf_{trans}	:	Largeur de la bande de transition
Δ_{ond}	:	Amplitude des ondulations en dB
ADN	:	Acide Désoxyribonucléique
AE	:	Algorithme Evolutionnaire
AG	:	Algorithme Génétique
AGC	:	Algorithme Génétique Canonique
AGH	:	Algorithme Génétique Hiérarchique
AGS	:	Algorithme Génétique Standard
AR	:	Autorégressif
ARMA	:	Autorégressif and moving average
ARN	:	Acide Ribonucléique
ASIC	:	Application Specefic Integrated Circuits
AttdB	:	Atténuation en dB
BMS	:	Bit le Moins Significatif
CAN	:	Convertisseur Analogique Numérique
CNA	:	Convertisseur Numérique Analogique
DSP	:	Digital Signal Processing
FCB	:	Filtre Coupe Bande
FPb	:	Filtre Passe bas
FPB	:	Filtre Passe Bande
FPGA	:	Field Programmable Gate Arrea
FPH	:	Filtre Passe Haut
GAlib	:	Genetic Algorithm library
IA	:	Intelligence Artificielle
IEEE	:	Institute of Electrical and Electronics Engineers

MA	:	Moving Average
MATLAB	:	MATrix LABoratory
PC	:	Personnal Computer
RIF	:	Filtre à Réponse Impulsionnelle Finie
RII	:	Filtre à Réponse Impulsionnelle Infinie
RSB	:	Rapport Signal sur Bruit