UNIVERSITÉ DE STRASBOURG

N° d'ordre :

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

UdS – INSA – ENGEES

THÈSE

présentée pour obtenir le grade de

Docteur de l'Université de Strasbourg
Discipline : Instrumentation et microélectronique

par

François Schwartz

Méthodologie de conception de systèmes analogiques. Utilisation de l'inversion ensembliste.

Soutenue publiquement le 18 novembre 2010

Membres du jury

Directeur de thèse : M. Yannick Hervé, Maître de Conférences, Université de Strasbourg
Rapporteur externe : M. Luc Jaulin, Professeur d'université, ENSIETA de Brest
Rapporteur externe : M. Ian O'Connor, Professeur d'université, Ecole centrale de Lyon
Examinateur : M. Luc Hebrard, Professeur d'université, Université de Strasbourg
Examinateur : Mme. Noëlle Lewis, Maître de Conférences, Université de Bordeaux I
Examinateur : M. Jacques Michel, Maître de Conférences, Université de Strasbourg

La vie de l'homme dépend de sa volonté ;
sans volonté, elle serait abandonnée au hasard
Confucius

Remerciements

Une thèse de doctorat est un travail de recherche, d'analyse et de synthèse et finalement de rédaction, de longue haleine, - certes individuel -, mais qui ne peut aboutir et prétendre à un niveau d'excellence, sans le soutien éclairé, renouvelé et bienveillant de personnalités aux compétences scientifiques reconnues.

Ma gratitude va d'abord à mon directeur de thèse, Yannick HERVE pour m'avoir proposé un sujet s'ouvrant sur un domaine de recherche prometteur d'avenir, celui de l'univers de la modélisation et de la méthodologie de conception, aux thématiques variées et riches. Ses conseils et l'appui de ses compétences m'ont été d'une aide précieuse.

Mes vifs remerciements et ma profonde reconnaissance vont ensuite, à Jacques MICHEL. J'ai largement pu profiter de sa vaste culture scientifique et de l'étendue de ses connaissances. Qu'il soit remercié pour sa disponibilité, le temps qu'il a bien voulu me consacrer généreusement tout au long de ces années de travail.

Je ne peux oublier mon équipier Sun QING qui m'a fait bénéficier de son regard critique dans la rédaction de certains travaux. Merci pour sa disponibilité et sa gentillesse.

Mes remerciements vont encore à mon directeur de laboratoire de l'InESS, M. Daniel MATHIOT pour m'avoir ouvert largement les portes du laboratoire, toutes ces années, condition indispensable à la réalisation d'un travail aussi exigeant. Sa compréhension, la prise en compte de ma situation particulière, m'a permis de concilier mon activité d'enseignant et de doctorant.

Un grand merci également aux rapporteurs, Messieurs Luc JAULIN et Ian O'CCONOR ainsi qu'aux examinateurs, Madame Noëlle LEWIS et Monsieur Luc HEBRARD, pour avoir pu bénéficier de leurs compétences reconnues dans la validation de mes travaux.

Mes remerciements s'adressent, enfin, à mon entourage familial et en particulier à mon épouse Sandrine, exemplaire de compréhension, prenant sur elle, un certain nombre de tâches et d'obligations qui me revenaient normalement, me permettant ainsi de progresser sereinement dans mon travail malgré les doutes et passages à vide .Une pensée à mes parents et beaux-parents pour leur soutien autant moral que concret comme la garde des enfants. Que mes enfants Clara, Simon, et Léane me pardonnent ces moments d'absence ou de présence sans être à leur écoute. J'ai le sentiment d'avoir manqué les premiers pas de leur enfance et j'ai tant à rattraper.

Je vous remercie de votre attention.

Table des matières

Liste des figures

Liste des tableaux

Représentation typographique

Les crochets [ ] sont réservés pour représenter un encadrement (intervalles et pavés).

Les variables a, u, … ou fonctions f, g, … seront imprimées en caractères italiques.

Les vecteurs v ou les fonctions vectorielles f seront représentées en caractères gras.

Ensembles

$\varnothing$	:	ensemble vide
A	:	ensemble
R	:	ensemble de tous les nombres réels
IR	:	ensemble de tous les intervalles de nombres réels
L	:	liste

Intervalles

$[a]$	:	intervalle contenant la variable *a*
a^-	:	borne inférieure de l'intervalle *a*
a^+	:	borne supérieure de l'intervalle *a*
$w([a])$	:	largeur de l'intervalle *a*
$[A]$	:	pavé
$[a_i]$	:	composante *i* du pavé [*A*]
$[A]_k$	:	pavé [*A*] à l'itération *k*
$[f]$	:	fonction d'inclusion de *f*
$[f]^*$	:	fonction d'inclusion minimale de *f*
A^{INNER}	:	réunion de pavés intérieurs

Autres symboles

$\triangleq$	:	égalité par définition
$\wedge$	:	ET logique
$\vee$	:	OU logique
$\times$	:	produit cartésien de deux ensembles
$\cup$	:	union d'ensembles au sens topologique
$\cap$	:	intersection d'ensembles
$\forall$	:	pour tous les nombres
$\exists$	:	existe un nombre
$\Rightarrow$, :=	:	opérateur d'affectation d'une valeur à une variable

Publications scientifiques

SCHWARTZ F., SUN Q., MICHEL J., HERVE Y., "A Robustness-Oriented Design Tool for the Topology Selection in Analog Synthesis", XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuits Design (SM2ACD'10), Tunisia, October 5-6, 2010.

SUN Q., MICHEL J., SCHWARTZ F., HERVÉ Y., « Implémentation d'un algorithme d'apprentissage par renforcement dans un réseau neuronal impulsionnel pour l'aide à la décision dans un pacemaker auto-adaptatif », XIIIèmes Journées Nationales du Réseau Doctoral de Microélectronique (JNRDM'2010), Montpellier (France), June 7-9, 2010.

SUN Q., SCHWARTZ F., MICHEL J., HERVÉ Y., "A reinforcment learning algorithm used in analog spiking neural network for an adaptive cardiac resynchronization therapy device", IEEE International Symposium on Circuits and Systems (ISCAS 2010), Paris (France), May 30 - June 2nd, 2010, Proc. pp. 2546-2549.

SUN Q., MICHEL J., SCHWARTZ F., « Étude sur l'implémentation d'un réseau neuronal impulsionnel analogique pour l'aide à la décision dans un pacemaker auto adaptatif », XIIèmes Journées Nationales du Réseau Doctoral de Microélectronique (JNRDM'2009), Lyon (France), May 18-20, 2009.

MICHEL J., SCHWARTZ F., « Analogue circuit sizing method using interval analysis", Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA'08), Montréal (Canada), June 22-25, 2008, Proc. pp. 331-334.

SCHWARTZ F., SUN Q., MICHEL J., "An OTA sizing method using interval analysis", XXIIIth Conference on Design of Circuits and Integrated Systems (DCIS'08), Grenoble (France), November 12-14, 2008.

SUN Q., SCHWARTZ F., MICHEL J., ROM R., "Implantation study of an analog spiking neural network in an auto-adaptive pacemaker", Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA'08), Montréal (Canada), June 22-25, 2008, Proc. pp. 41-44.

SCHWARTZ F., MICHEL J., « Modélisation d'un réseau de neurones sous MATLAB et en VHDL-AMS : Application à la reconnaissance de caractères », 7ème Colloque sur le Traitement Analogique de l'Information, du Signal et ses Applications (TAISA'2006), Strasbourg (France), October 19-20, 2006.

Chapitre 1 - introduction générale

Parmi les nouvelles problématiques liées à la conception des applications en électronique, deux en particulier, font l'objet d'une attente forte. Face à l'abondance et à la diversité des informations résultant de sources nombreuses et différentes, la question se pose de savoir, quelles architectures électroniques proposer pour produire un maximum de puissance de traitement ? A ce premier problème s'ajoutent les difficultés techniques de l'intégration de systèmes complexes et intelligents sur un seul circuit (SSoC), que rencontrent les équipes de concepteurs ou les fabricants de CI. Ces difficultés entravent la progression du cycle de conception du produit et en retardent le temps de mise sur le marché.

Quelles sont précisément ces deux problématiques et les pistes pour les contourner ?

Le choix des architectures électroniques capables de produire un maximum de puissance de traitement.

La chaîne de traitement de l'information est classiquement composée d'un capteur, d'un dispositif de traitement et d'une partie opérative. La figure 1-1 en donne une illustration.

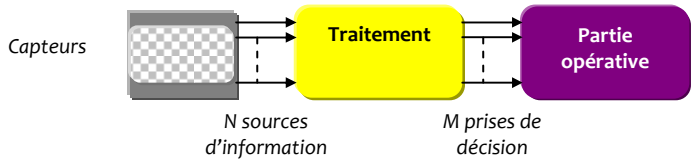


Figure 1-1 Synoptique d'une chaîne de traitement de l'information

Les actions d'acquisition, de traitement et de décision doivent être prises dans des délais courts à partir d'une masse importante de données captées.

Comment, par exemple, assister un conducteur de voiture en donnant en temps réel la localisation précise du véhicule par rapport aux bords de la voie ? Cette opération nécessite de fortes capacités de calculs où à partir d'un flot d'informations particulièrement élevé provenant d'une caméra embarquée, un nombre limité de décisions telles que l'avertissement sonore ou encore la correction de la trajectoire sont attendues en un temps très court. On constate qu'à travers cet exemple, les fonctions acquérir, traiter et décider sont essentielles et vont permette de transformer des informations « brutes » en informations « validées » auxquelles sera associée une crédibilité maximale.

La multiplicité des sources d'information nécessite un dispositif de traitement parallèle. Dans cette classe de système on constate que les réseaux de neurones peuvent offrir une réponse adaptée.

Les réseaux de neurones artificiels comportent des unités de calculs parallèles fortement interconnectées qui réalisent des opérations non linéaires de leurs entrées vers leur(s) sortie(s). Ils permettent d'extraire le maximum d'informations pertinentes à partir d'une quantité de données dense et hétérogène dans de nombreuses situations telles que la prédiction de propriétés, la modélisation de fonctions, ou encore la reconnaissance de formes. De telles architectures sont donc capables de prévoir des évènements, expliquer des résultats observés, ou même résoudre des problèmes mal posés ou mal définis… L'universalité est une autre caractéristique des réseaux de neurones qui leur donnent la faculté de s'adapter dynamiquement à des situations diverses : ainsi, en généralisant son expérience passée, il sera capable de prendre une décision face à une situation inédite !

L'implémentation d'un réseau de neurones dans un circuit soulève de nouvelles problématiques liées, notamment, à la contrainte de surface et de la consommation énergétique. La solution neuronale numérique est la plus répandue et rassemble son architecture avec son algorithme d'apprentissage dans un circuit de grande complexité, typiquement un FPGA. Les grandes capacités d'intégration acquises à ce jour permettent d'implémenter en masse de telles fonctions dans cette famille de composants. Néanmoins, un traitement parallèle avec la multiplication des unités de calcul entraîne un accroissement de la surface du silicium du circuit. De plus, la quantité d'informations traitée à des cadences élevées pour assurer les opérations d'échantillonnage ou de conversion analogique-numérique ne contribue pas à une consommation énergétique réduite et à des temps de réponse optimisés. En revanche, la solution neuronale analogique concentre dans un volume réduit, une capacité de calculs élevée avec une minimisation du rapport énergie/capacité de traitement [Maa-97, Kak-09]. Si en plus, on y ajoute l'aspect temps réel alors les réseaux de neurones analogiques offrent une alternative très intéressante à leur contrepartie numérique.

Leur mise en œuvre est néanmoins longue et délicate (aspect non-idéal des composants) et nécessite des moyens méthodologiques qui s'inscrivent dans la problématique générale des nouvelles méthodes de conception et de synthèse des systèmes.

Les difficultés techniques de l'intégration de systèmes complexes et intelligents sur un seul circuit.

La pression du marché des circuits de l'électronique connaît une évolution considérable. Cette pression pousse les concepteurs de circuits intégrés à se tourner vers des processus de conception fiables pour lesquels le futur produit est assuré d'être mis sur le marché dans des délais les plus courts et prédéterminés. La garantie de la maîtrise du temps de mise sur le marché (*Time To Market*) du produit devient un facteur crucial de réussite pour le concepteur. Si la conception de solutions purement numériques est à ce jour exécutée par des outils qui assurent ces exigences, il n'en est pas de même lorsqu'une conception mixte analogique/numérique est envisagée. Un circuit électronique analogique requiert généralement un nombre d'itérations de sa phase de conception qui peut devenir élevé et donc pénalisant pour le concepteur. Au mieux, une solution peut être trouvée mais dans des délais difficilement quantifiables, au pire, le processus de conception itère indéfiniment sans proposer de solution satisfaisante au regard des spécifications attendues ! Il en résulte par conséquent une prise de risque et un coût industriel qui deviennent trop importants.

Le développement d'un circuit nécessite des environnements très différents à mesure que le projet progresse dans sa phase de conception. Les outils ne seront pas les mêmes selon qu'il s'agisse d'une analyse de haut niveau (outil MATLAB par exemple), d'une analyse de bas niveau (SPICE) [Don-94]. Dans le but d'accélérer la phase de conception, la solution communément proposée est d'automatiser le processus de dimensionnement [Gie-05a, Lop-06, Rut-07]. Cette prise de conscience a amené les constructeurs à augmenter le taux de réutilisation des composants virtuels IP (*Intellectual Property*) d'un projet [Mil]. Or, le concept de réutilisation ne peut être généralisé au travers de tous les niveaux hiérarchiques qu'à la condition de disposer d'un environnement qui permette de rassembler l'ensemble des besoins (cahier des charges, spécifications) et des fonctionnalités du circuit dans un même outil de description. Le langage VHDL-AMS propose à ce titre des moyens pour y répondre. Le langage VHDL-AMS permet d'appliquer plusieurs concepts méthodologiques fondamentaux [Her-02]. Parmi ceux-là on trouve, la possibilité d'exprimer des éléments du cahier des charges en entités simulables, une conception descendante (top-down), une description multi-abstraction, une simulation environnementale, la réutisabilité de composants ou d'expériences, etc…

Les pistes de notre action…

Notre démarche s'articule autour de deux thématiques fortes traitées en cinq chapitres où nous cherchons :

- à proposer des architectures électroniques capables de produire un maximum de puissance de traitement de l'information.
- à en minimiser le temps de conception.

Nous développerons notre démarche autour d'un démonstrateur qui sera constitué d'un réseau de neurones analogiques. Ce réseau sera entraîné à la fonction de reconnaissance de caractères qui est une application largement répandue dans le domaine de l'intelligence artificielle. Cette application met en œuvre un processus non déterministe que les réseaux de neurones sont particulièrement aptes à reproduire. L'idée d'implémenter dans un circuit analogique l'architecture du réseau et son algorithme d'apprentissage permettra de souligner les difficultés et contraintes liées à la conception de circuits électroniques. Cela permettra surtout de proposer des pistes afin de soutenir notre démarche.

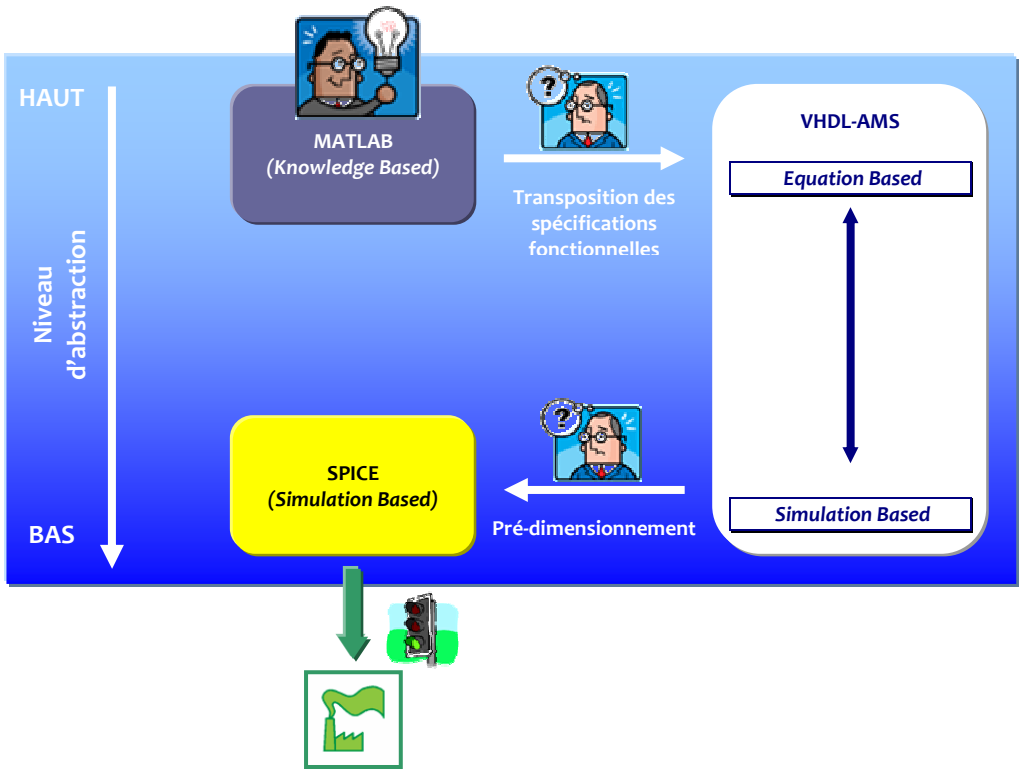


Figure 1-2 Positionnement de notre démarche dans le cycle de conception d'une application

Le chapitre 2 fera la présentation d'une architecture particulière à réseau de neurones et des critères de choix qui ont guidés la conception architecturale. Il s'agira d'évaluer la faisabilité de la configuration la plus adaptée à répondre aux contraintes de traitement des flux importants d'informations. Ce travail d'expertise est mené à partir d'une base de connaissance (Knowledge Base) et se positionne au début du cycle de conception. La figure 1-2 en donne une illustration. Le chapitre 3 proposera un partitionnement du circuit neuronal en blocs élémentaires selon un processus de décomposition classique. Il s'agira d'abord de transposer les spécifications fonctionnelles élaborées sous MATLAB dans le langage VHDL-AMS. Ce passage est délicat et implique la mise en place d'une procédure de validation. Il s'agira ensuite d'une part de

déterminer le cahier des charges de chaque bloc tout en garantissant la compatibilité avec l'application et d'autre part, de minimiser le temps de développement de chaque bloc puis celui de l'ensemble du réseau. Partant du postulat que le temps de développement du système (réseau de neurones) est fonction du temps de développement de chaque bloc et du nombre d'itérations nécessaires aux niveaux hiérarchiques supérieurs (adéquation performances – spécifications), on souhaite alors introduire le principe de « garantie » des solutions. Cela nous a orienté vers l'utilisation des calculs par intervalles pour le dimensionnement des blocs et à l'utilisation conjointe du Prototypage Virtuel pour la détermination du cahier des charges de chaque bloc dans le système. Ce dernier point vise à répondre à la problématique de formalisation du cahier des charges et s'appuie sur les propriétés du langage VHDL-AMS. Le chapitre 3 montrera la démarche méthodologique qui fut la notre dans l'analyse du réseau de neurones.

Les chapitres 4 à 6 porteront plus spécifiquement sur le dimensionnement d'une architecture d'un bloc élémentaire.

Le chapitre 4 présentera en particulier les limites des techniques actuelles en matière de synthèse analogique et un état de l'art sera par ailleurs présenté. Les calculs par intervalles seront ensuite introduits.

Le chapitre 5 fera une présentation plus détaillée des méthodes de dimensionnement dédiées à l'exploration de l'espace de conception en analyse par intervalles. On introduira aussi une spécificité, les contracteurs, qui permettra de compenser une limite de cette nouvelle arithmétique. Une présentation de la démarche méthodologique basée sur l'analyse par intervalles sera illustrée à partir d'exemples académiques. Dans le chapitre 6, les méthodes de dimensionnement en analyse par intervalles seront appliquées sur des circuits de l'électronique. Ces exemples s'appuient sur des références classiques de circuits à amplificateurs à transconductance (OTA) qui jouent un rôle d'interface entre les entrées et les fonctions internes au neurone. Nous montrerons que la démarche méthodologique est assez générale afin de pouvoir ensuite être appliquée aux fonctions neuronales.

Chapitre 2 - choix d'une architecture parallèle à réseau de neurones

2.1 Introduction

La vitesse de traitement des informations élémentaires dans le cerveau humain est relativement lente et résulte d'un processus physico-chimique complexe. Le cerveau est pourtant capable de réussir en peu de temps des opérations extrêmement complexes comme la reconnaissance des visages. Avec près de 40 milliards de neurones reliés en réseau, la réussite de telles performances est le fait d'un traitement distribué et parallèle de l'information [Abd-94]. Aussi, un certain nombre de modèles artificiels fortement inspirés des neurones biologiques ont été inventés pour accomplir des tâches de calculs dans des réseaux. Il existe à ce jour plusieurs familles de réseaux de neurones que l'on classe selon leur topologie mais aussi en fonction de leur méthode d'apprentissage. Chacune d'entre elles trouve un champ d'application (classification, identification, …) et seront succinctement présentées dans ce chapitre.
Nous ferons alors le choix d'une architecture particulière en fonction de critères liés à la tâche de classification. La classification est une opération délicate qui n'a pas de formalisme explicite et révèle une part de la complexité des problèmes actuels. Associer cette opération à une topologie d'un réseau permettra de définir le prototype d'étude à notre démarche de conception. C'est l'objectif de ce chapitre qui sera présenté en deux parties. Dans la première partie, une synthèse des différentes générations de modèles de neurones ainsi que des principales topologies de réseaux de neurones sera proposée. Dans la seconde partie, les méthodes d'apprentissage seront illustrées dans le cadre de la mémorisation par un réseau de neurones d'une équation logique élémentaire. Ceci permettra en fin de chapitre de statuer sur la topologie et la méthode d'apprentissage la plus adaptée à la tâche de la reconnaissance de caractères par un réseau de neurones implémentés dans un circuit analogique.

2.2 Les réseaux de neurones

Nous allons décrire brièvement, dans cette partie, les différentes architectures à réseaux de neurones. Nous allons d'abord nous intéresser à l'élément de base d'un réseau qu'est le neurone puis une synthèse des différentes générations de neurones sera proposée. L'étape suivante nous conduira à l'étude des réseaux de neurones.

2.2.1 Le neurone

Les neurones sont des unités de calcul qui génèrent un signal de sortie en fonction des signaux d'entrée. Chaque neurone est stimulé sur ses entrées par des signaux dont l'intensité est pondérée par des coefficients que l'on appelle « poids synaptiques » du neurone. Les signaux d'entrée pondérés sont sommés pour donner une grandeur appelée « potentiel d'activation ». Ce potentiel d'activation est ensuite appliqué à une fonction de seuil appelée « fonction d'activation » du neurone. La figure 2-1 donne une représentation simplifiée d'un neurone à *n* entrées et à fonction d'activation Φ.

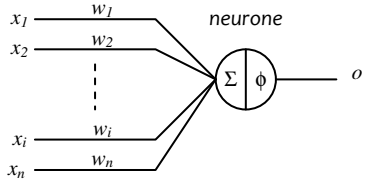


Figure 2-1 Représentation d'un neurone à n entrées (x_1 à x_n) et à fonction d'activation Φ. A chaque entrée est associée un poids synaptique (w_1 à w_n).

Le potentiel d'activation du neurone s'écrit :

$$a = \sum_{i=1}^{n} w_i x_i$$

Avec : - n : le nombre total d'entrées du neurone
- x_i : l'entrée i du neurone
- w_i : le poids synaptique i du neurone

La sortie du neurone est donc une fonction de son potentiel d'activation :

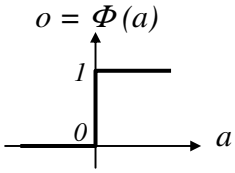
$$o = \Phi(a) = \Phi(\sum_{i=1}^{n} w_i x_i)$$

Les premiers modèles de neurones étaient fondés sur des sorties à réponses binaires et synchrones dont la fonction d'activation était une fonction échelon. La seconde génération de modèles construisait une réponse nuancée et toujours synchrone. La fonction d'activation était une fonction continue. La troisième génération enfin, plus récente, utilise des neurones qui génèrent des impulsions sur leurs sorties en asynchronisme avec les stimuli. Nous allons les présenter.

2.2.1.1 Première génération : Les modèles de neurones à fonction échelon

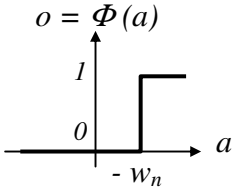
La première génération de neurones était basée sur le modèle de *McCulloch-Pitts* [Dav-94]. Ce modèle utilise la fonction d'activation à seuil telle que :

$$\begin{cases} o = \Phi(a) = 0 & si\ a \le 0 \\ o = \Phi(a) = 1 & si\ a > 0 \end{cases}$$



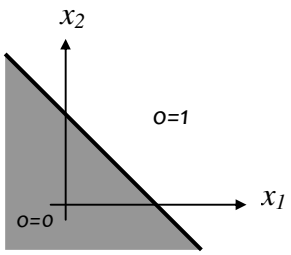
Le seuil est défini par défaut à la valeur nulle mais peut bien entendu être modifié en figeant une entrée. Si l'on fixe par exemple l'entrée n du neurone à la valeur x_n =1, le poids synaptique associé w_n définira un nouveau seuil de telle sorte que :

$$\begin{cases} o = \Phi(a) = 0 & si\ x_1 w_1 + ... + x_{n-1} w_{n-1} \le -x_n w_n = -w_n \\ o = \Phi(a) = 1 & si\ x_1 w_1 + ... + x_{n-1} w_{n-1} > -x_n w_n = -w_n \end{cases}$$



L'entrée x_n provoque dont un décalage du seuil de la fonction d'activation. On nomme généralement cette entrée, l'entrée de « biais » du neurone qui sera désignée par la lettre b. Le biais peut être considéré comme la pondération de l'entrée b fixée à 1.

Le cas d'un modèle à deux entrées permet de mettre en évidence les limites d'un tel neurone. En effet, la sortie détermine, dans un plan de ces entrées, deux régions séparées par la droite : $x_1 w_1 + x_2 w_2 + b = 0 \quad \Leftrightarrow \quad x_2 = -(w_1 / w_2).x_1 + b / w_2$.



Par conséquent, seuls des problèmes de classification avec des classes linéairement séparables peuvent être résolus avec ce type de neurone (cas d'une porte OU par exemple).

Exemple : Dispositif d'aide à la conduite

Cet exemple propose simplement de mettre en évidence le mécanisme décisionnel d'un neurone à fonction échelon.

Supposons un système rudimentaire d'assistance à la conduite constitué d'un réseau à un neurone à trois entrées. Ce réseau devra activer une alarme sonore en plaçant l'état de sa sortie à l'état booléen 1 en cas de détection d'une situation critique.
Chacune des entées renseigne le réseau de l'état de conduite du véhicule de la façon suivante :

- l'entré x_1 sera à l'état booléen 1 si la vitesse du véhicule est supérieure à 100km/h et 0 dans le cas contraire. Cet indicateur ne doit pas provoquer à lui seul l'activation du signal d'alarme. Le poids synaptique associé sera donc de valeur faible, nous prendrons $w_1 = 0{,}5$.
- l'entré x_2 sera à l'état booléen 1 si la présence d'un véhicule est détectée à moins de 10m et 0 dans le cas contraire. Cet indicateur ne doit pas provoquer à lui seul l'activation du signal d'alarme mais uniquement en cas de vitesse du véhicule supérieure à 100km/h. Le poids synaptique associé sera donc de valeur faible, nous prendrons également $w_2 = 0{,}5$.
- l'entré x_3 sera à l'état booléen 1 en cas de manifestation d'hypovigilance du conducteur et 0 dans le cas contraire. Cet indicateur doit provoquer à lui seul l'activation du signal d'alarme. Le poids synaptique associé sera donc de valeur forte, nous prendrons $w_1 = 1{,}5$.

En y ajoutant le poids *b* de l'entrée de biais, le potentiel d'activation s'écrit de la façon suivante :

$$a = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

Le neurone répondra par un état 1 dans le cas où :

$$x_1 w_1 + x_2 w_2 + x_3 w_3 > -b$$

C'est à dire plus précisément lorsqu'au moins un des deux cas suivants se manifeste :

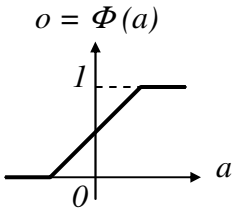
- $x_1 = 1$ ET $x_2 = 1$; dans ce cas : $x_1 w_1 + x_2 w_2 + x_3 w_3 = w_1 + w_2 = 1.0$
- $x_3 = 1$; dans ce cas : $x_1 w_1 + x_2 w_2 + x_3 w_3 = w_3 = 1.5$

Il alors nécessaire de fixer le biais à une valeur comprise entre 0.5 et 1.0 qui correspondent respectivement à la manifestation d'une des deux premières entrées et le neurone doit répondre par *o = 0*, ou des deux simultanément et dans ce cas *o = 1*. En prenant la valeur médiane par exemple, il vient : *b* = -0.75.

Cet exemple illustre une situation permettant d'identifier deux régions distinctes et séparées par un plan pour lesquelles un cas de décision booléen peut être pris à partir d'un neurone à fonction échelon. De fait, la simplicité du modèle de première génération a fait place au modèle de seconde génération dont les caractéristiques non linéaires permettent d'augmenter les performances de classification.

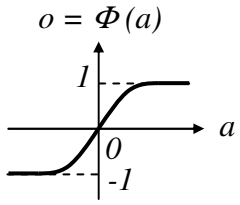
2.2.1.2 Seconde génération : Le neurone à fonction d'activation continue

La seconde génération de neurones est basée sur des unités de calculs qui construisent une réponse continue résultant de la somme pondérée des entrées. Les fonctions d'activation sont classiquement les fonctions linéaires saturées ou sigmoïdales :



Fonction d'activation linéaire saturée :

$$\begin{cases} \Phi(a) = a & si -0.5 \le a \le 0.5 \\ \Phi(a) = 0 & si\ a < -0.5 \\ \Phi(a) = 1 & si\ a > 0.5 \end{cases}$$



Fonction d'activation sigmoïde :

$$\Phi(a) = \tanh(a)$$

On peut considérer que la fonction linéaire saturée est une restriction de la fonction sigmoïdale et en particulier de sa partie linéaire. La fonction linéaire saturée n'a pas d'utilité pratique (son utilisation se limite elle aussi aux classes de problèmes linéairement séparables) même si des exemples à vocation pédagogique y font souvent référence.

Les neurones issus de la seconde génération sont capables de calculer des fonctions à partir de valeurs non booléennes (possibilité d'avoir des réponses intermédiaires). Ces neurones comportent des fonctions d'activation qui peuvent être dérivables ce qui, nous le verrons plus loin, présente un intérêt crucial dans l'apprentissage des réseaux multicouches.

2.2.1.3 Troisième génération : Le neurone impulsionnel

Le modèle du neurone impulsionnel résulte d'une représentation biologique plus réaliste en reprenant l'idée d'une transmission de l'information sous la forme d'impulsions électriques entre neurones. L'aspect temporel contribue directement à la génération de la réponse d'un neurone. Le modèle « *Integrate and Fire* » est par exemple un modèle couramment employé. Il construit sa réponse en cumulant l'ensemble des impulsions qui convergent vers ses entrées dont chacune est pondérée par un poids synaptique. La mesure du cumul des impulsions résulte d'une intégration qui est ensuite comparée à un seuil. Un dépassement de ce seuil provoque alors la génération d'une nouvelle impulsion après un délai déterminé. La représentation d'un neurone impulsionnel est donnée sur la figure 2-2 avec un chronogramme illustrant un scénario d'excitation.

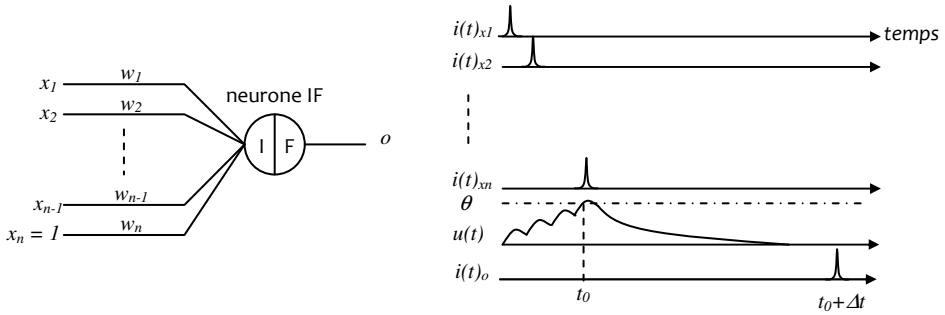


Figure 2-2 Représentation d'un neurone impulsionnel et d'un scénario d'excitation. Les courants impulsionnels de chaque entrée i(t)xi sont intégrés pour donner le signal u(t). Le neurone délivre une nouvelle impulsion de courant i(t)o.

Les réseaux à neurones impulsionnels nécessitent moins de neurones que leurs homologues à fonction d'activation continue [Maa-97]. Ceci fait de ce type de neurone, une unité de calcul bien plus puissante permettant de traiter l'information dans des temps plus courts [Kun-02]. Par ailleurs, l'utilisation de réseau de neurones impulsionnels se généralise à de plus en plus d'applications [Sun-08].
Néanmoins, un tel réseau a besoin de circuits annexes pour générer les trains d'impulsions mais aussi pour réaliser les fonctions internes aux neurones (intégration, seuillage, génération d'impulsions). Pour cette raison, nous limiterons notre étude au champ des neurones à fonction d'activation continue.

Nous allons maintenant faire un rapide tour d'horizon des structures des réseaux de neurones avant de proposer une architecture et son processus d'apprentissage adaptés à la fonction spécifique de reconnaissance de formes.

2.2.2 *Les principales topologies de réseaux de neurones*

Un réseau de neurones résulte d'une interconnexion organisée des neurones entre eux. Cette organisation fait apparaître principalement trois topologies de connexions différentes : les réseaux bouclés, les réseaux topologiques et les réseaux non bouclés. Nous allons maintenant vous les présenter.

2.2.2.1 Les réseaux bouclés ou récurrents

Les réseaux bouclés se caractérisent par la présence d'au moins une boucle de rétroaction c'est-à-dire le renvoi d'au moins une sortie du réseau vers l'une de ses entrées. Les neurones sont à sortie binaire. La figure 2-3 en donne une représentation avec l'apparition de fonctions à retard unitaire qui introduisent des échantillons d'un état précédent de la sortie du réseau sur son entrée. La sortie d'un réseau bouclé peut donc être fonction d'elle-même et l'aspect temporel est par conséquent explicitement pris en considération. Ces réseaux réalisent donc des équations aux différences adaptés notamment à l'identification de processus dynamiques (modélisation, commande de processus, …) [Dre-04].

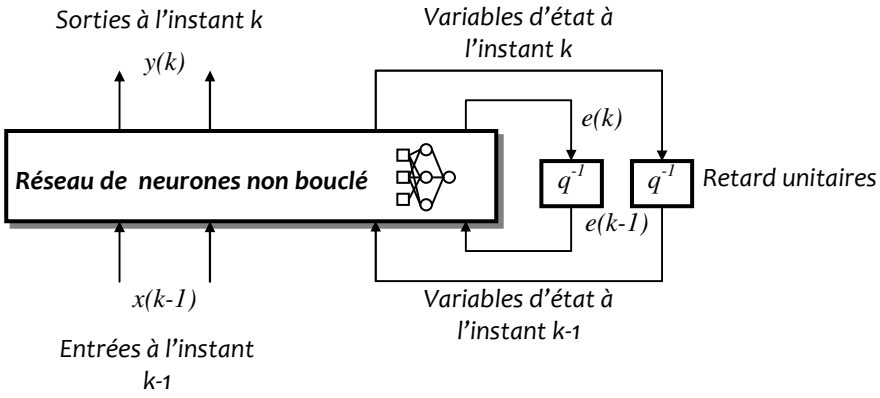


Figure 2-3 Représentation fonctionnelle d'un réseau bouclé.

Ce type de réseau est cependant difficile à mettre en œuvre et, vu le caractère récurent des états du réseau sur ses entrées, les problèmes de convergence sont plus fréquents au sens où il n'est généralement pas possible de garantir la convergence vers un point fixe [Gui-02]. Les réseaux bouclés sont également plus consommateurs de temps de calcul que leur équivalent les réseaux non bouclés [Bon-05].
La force de telles architectures est de pouvoir associer une donnée présentée au réseau, même incomplète, à une donnée qui a été préalablement mémorisée. On parle alors de mémoire associative ; chaque donnée mémorisée crée un bassin d'attraction et toute présentation partielle de motifs, en entrée du réseau, entraîne le réseau dans le bassin le plus proche de l'entrée. L'identification du bassin vers lequel le réseau a convergé permet d'associer l'entrée à une donnée mémorisée.

2.2.2.2 Les cartes topologiques

Il s'agit de réseaux de même type que les réseaux non bouclés mais dont l'apprentissage est à compétition. Typiquement, on souhaite que seul un neurone soit actif dans une région donnée du réseau (cas de l'apprentissage WTA [Wil-03]). Les neurones placés au plus près des stimuli renforcent ainsi leur attractivité. La carte « synaptique » s'organise alors en fonction de ces excitations. La figure 2-4 illustre le cas d'un réseau organisé en carte connu sous le nom de « cartes de Kohonen », dont le principe consiste à trouver une projection entre l'espace des données d'une grande dimension et l'espace des représentations à une dimension plus réduite. La projection doit alors conserver la topologie des données.
L'espace de sortie est un espace dont on peut alors représenter les données de l'espace d'entrée pour de grandes dimensions. Ce type de réseau se révèle être particulièrement adapté au prétraitement des données en décrivant des observations telle une analyse en composantes principales non linéaire.

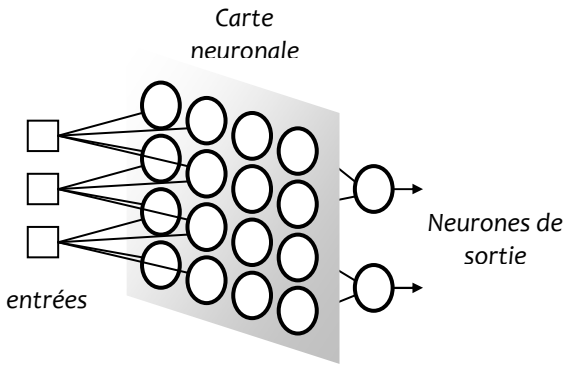


Figure 2-4 Réseau de neurones à carte topologique avec représentation partielle de la couche d'entrée.

Les réseaux à cartes topologiques souffrent néanmoins d'un certains nombre de défauts. Leur forme étant fixée a priori, un grand nombre d'unités ne pointant pas dans la distribution d'entrée conduit à les rendre inactives [Her-94]. Le temps de convergence des réseaux à cartes topologiques est important pour représenter au mieux la topologie de l'espace de départ. En effet, pour assurer la convergence, il est nécessaire de présenter récursivement les motifs de l'espace d'entrée et de préférence de façon aléatoire pour contribuer à une meilleure qualité d'apprentissage. Or, dans un contexte réel et d'utilisation en parfaite autonomie, de telles conditions sont difficiles à satisfaire. De ce fait, les cartes topologiques sont inadaptées à accomplir des opérations de classement de formes mais peuvent en revanche parfaitement s'appliquer à des opérations de prétraitements.

2.2.2.3 Les réseaux non bouclés ou feedforward

Dans le cas des réseaux de neurones non bouclés, l'information circule des entrées vers les sorties sans «retour en arrière» avec une organisation des neurones en couches successives. Pour ce type de structure, chacune des entrées est reliée à chaque neurone de la première couche appelée couche cachée. A leur tour, chacune des sorties des neurones de la première couche est reliée à l'ensemble des neurones de la couche suivante et cette règle se généralise jusqu'à atteindre la dernière couche appelée couche de sortie. La figure 2-5 donne la représentation graphique d'une topologie de réseau non bouclé à une couche d'entrée (couche sans neurone), une couche de neurones cachée et une couche de neurones de sortie.

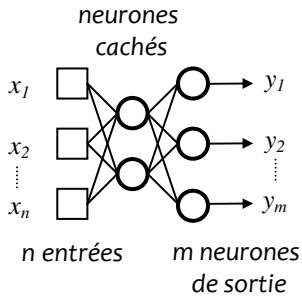


Figure 2-5 Réseau de neurones non bouclé à une couche cachée et une couche de sortie.

Le fait que l'information soit propagée directement de la couche d'entrée vers la couche de sortie rend en théorie le réseau sans délai et par conséquent, le temps ne joue aucun rôle fonctionnel dans un réseau non bouclé.

Ce type de réseau ne peut donc réaliser qu'une fonction mathématique classique (linéaire ou non linéaire) de la forme :

$$y_s = \Phi[\sum_{i=1}^{n} w_i x_i \,] \qquad \text{avec : } s=1,\ldots,m$$

Les réseaux de neurones non bouclés sont par conséquent utilisés pour des transformations non linéaires (mapping) d'une variable d'entrée multidimensionnelle en une autre variable de sortie qui peut elle aussi être à plusieurs dimensions. Il réalise donc un « mapping » d'un espace de dimension *n* vers un espace de dimension *m*. Ce genre de réseau possède une architecture universelle qui réalise en théorie n'importe quelle transformation de ses variables d'entrée [Wil-03]. En pratique, cette transformation est réalisable si les neurones en couche caché sont en nombre suffisant et il n'existe pas aujourd'hui de méthode précise pour y parvenir. En général, plus grand sera la nombre de neurones en couche cachée, plus il sera possible de réaliser des transformations complexes.

2.2.3 Choix de la topologie

Nous rappelons que l'objectif est de retenir l'architecture qui permette de traiter en parallèle une quantité de données en temps réel. Le réseau doit être de complexité limité afin de pouvoir mettre en œuvre au plus vite notre méthode de conception. Mais il devra aussi être suffisamment développé en terme de fonctions à concevoir afin d'évaluer la capacité de la méthode à être généralisée à des applications plus complexes. Cet équilibre entre matière à concevoir et matière à expertiser nous mène à nous orienter vers un réseau dans sa plus simple expression c'est-à-dire bâti autour d'un unique neurone. Un neurone à lui seul comporte des fonctions de somme et de seuil et la fonction de multiplication sur ses branches synaptiques. Par ailleurs, en reprenant les critères de consommation et de surface, le démonstrateur doit être un circuit de taille mesurée. Parmi les structures précédemment présentées, il parait alors judicieux de se tourner vers une architecture de type *feedforward*. Ce choix nous permet ainsi d'orienter le travail du réseau vers la tâche de classification et les réseaux non bouclés y sont particulièrement efficaces. En revanche, les réseaux récurrents sont davantage utilisés pour modéliser des comportements dynamiques complexes et pour reproduire - apprendre - ces comportements. Enfin, l'atout d'un temps d'exécution court pèse également dans cette décision.

2.3 Apprentissage d'un réseau de neurones

Un réseau de neurones est caractérisé par ses paramètres appelés poids synaptiques sur lesquels l'algorithme d'apprentissage agit en les modifiant afin que le réseau remplisse au mieux la mission qui lui est confiée.

La phase d'apprentissage constitue la première des trois situations d'utilisation d'un réseau de neurones. En effet, trois phases peuvent être envisagées dans le cycle de vie d'un réseau de neurones dans l'ordre suivant :

① Phase d'apprentissage du réseau de neurones : les poids synaptiques du réseau sont ici corrigés en leur donnant au départ du processus d'apprentissage une valeur aléatoire. Un jeu de vecteurs est ensuite présenté à l'entrée du réseau, et lorsque le niveau d'entraînement du réseau est jugé suffisant, le processus d'apprentissage est stoppé. La seconde situation est alors effectuée.

② Phase de validation du réseau de neurones : Un jeu de vecteurs différent du précédent est présenté à l'entrée du réseau. Cette phase consiste à mesurer la robustesse du réseau et donc à évaluer la qualité de l'apprentissage. Dans le cas où le réseau passe le test de

validation avec succès, le réseau peut ensuite être mis en exploitation. Dans le cas contraire, une nouvelle phase d'apprentissage doit être envisagée.

③ Phase d'exploitation du réseau de neurones : Le réseau rentre en phase d'exploitation et est placé dans son environnement applicatif.

La phase d'apprentissage est maintenant présentée plus en détails. Des exemples seront ensuite proposés plus loin pour notamment illustrer le cas de la phase de validation.

On distingue deux types d'apprentissage : l'apprentissage supervisé et non supervisé. Un apprentissage est dit supervisé lorsque l'on force le réseau à converger vers un état final précis (cible), en même temps qu'on lui présente un motif (vecteur d'entrée). À l'inverse, lors d'un apprentissage non-supervisé, le réseau est laissé libre de converger vers n'importe quel état final lorsqu'on lui présente un motif.

Il existe deux processus de mise à jour des poids synaptiques. Ces deux processus dépendent de la façon dont sont présentés les vecteurs d'entrée au réseau en phase d'apprentissage :

Le processus d'apprentissage est dit « on-line » ou en mode incrémental lorsqu'on met à jour les poids pour chaque échantillon d'apprentissage présenté au réseau de neurones.
Le processus d'apprentissage est dit « off-line » ou en mode batch lorsqu'on calcule d'abord les erreurs pour tous les échantillons sans mettre à jour les poids (on additionne les erreurs c'est-à-dire l'écart entre la sortie du réseau et de la cible) ; lorsque l'ensemble des données est passé une fois dans le réseau, on applique la correction en relation avec l'erreur totale qui a été cumulée.

Le principe des apprentissages non-supervisé et supervisé sont maintenant présentés :

2.3.1 Apprentissage non supervisé

Un réseau de neurones non bouclé peut être utilisé dans un but de visualisation ou d'analyse de données. On dispose d'un ensemble de données, représentées par des vecteurs de grande dimension, et l'on cherche à les regrouper, selon des critères de ressemblance qui sont inconnus *a priori*. On parlera alors de classification, d'extraction de composantes principales, etc… La figure 2-6 illustre le principe de l'apprentissage non supervisé.

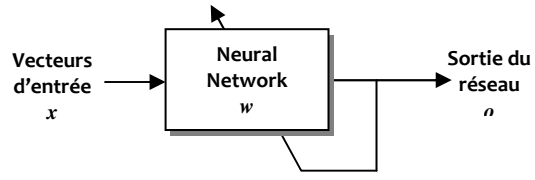


Figure 2-6 Système à apprentissage non supervisé.

Il n'y a donc pas d'intervention extérieure. Les réseaux récurrents (avec la règle de Hebb) ou à carte topologique (apprentissage compétitif tel que WTA) utilisent typiquement ce type d'apprentissage.

2.3.2 Apprentissage supervisé

Les données d'apprentissage sont composées d'un ensemble de vecteurs d'entrée ou « patterns » x appliqué au réseau et d'un ensemble de données correspondant à l'état de la sortie attendue par le réseau appelées cibles ou « targets » t. La sortie o du réseau de neurones est donc une fonction f_{RN} des variables d'entrées (les patterns) et des poids synaptiques w :

$$o = f_{RN}(w,x)$$

La fonction du réseau $f_{RN}(w,x)$ doit donc approcher une fonction théorique généralement non-linéaire $f_{NL}(x)$.On connaît donc, en tous points ou seulement en certains points, les valeurs que doit avoir la sortie du réseau en fonction des entrées correspondantes. On parlera donc d'apprentissage supervisé. La figure 2-7 en schématise le principe.

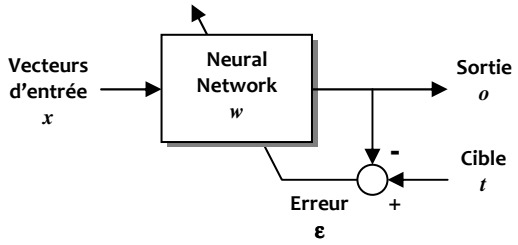


Figure 2-7 Système à apprentissage supervisé.

Les poids synaptiques, de chacun des neurones, sont donc ajustés de manière à faire tendre la sortie du réseau *o* vers la valeur cible *t* :

$$|o - t| \rightarrow 0 \text{ c'est-à-dire : } f_{RN}(w,x) \rightarrow f_{NL}(x)$$

Dans ce cas, la fonction du réseau $f_{RN}(w,x)$ est équivalente à la fonction à modéliser $f_{NL}(x)$ [AF55] et :

$$f_{RN}(w,x) \cong f_{NL}(x)$$

Ces poids sont évalués au travers d'un algorithme d'apprentissage. On retrouve en autre ces algorithmes sous le nom de LMS (Widrow et Hoff), « *Back Propagation* » (BP) ou l'algorithme dit de perturbation des poids « *Weight Perturbation* » (WP). Chacun ajuste la valeur des poids synaptiques selon l'évolution du gradient du signal d'erreur défini par la différence entre la valeur de la sortie attendue et celle du réseau.

2.3.3 Exemples d'apprentissages supervisés

Trois règles d'apprentissage sont présentées : la règle de Widrow-Hoff, de la rétropropagation du gradient et du Weight Perturbation. La règle de Widrow-Hoff se limite à une correction des poids synaptiques des réseaux à une seule couche de neurones. Les règles de la rétropropagation du gradient et du Weight Perturbation élargiront leur champ d'action aux réseaux à plusieurs couches de neurones.

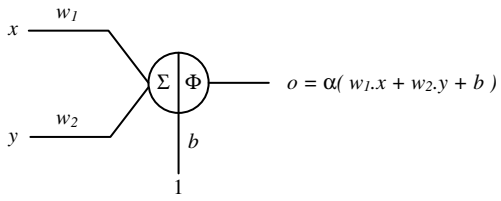
2.3.3.1 Règle de Widrow et Hoff (LMS) [Hui-94]

L'algorithme LMS s'applique aux réseaux dont les neurones sont à fonction d'activation linéaire :

$$\Phi(a) = \alpha.a$$

Où α désigne le coefficient directeur de la fonction d'activation.

Considérons le réseau à un seul neurone suivant :



A chaque entrée appliquée au réseau, la sortie du réseau o est comparée à la valeur souhaitée t (vecteur cible). L'erreur ε est calculée par la différence entre la valeur cible et la sortie du réseau (figure 2-7)

En mode incrémental, l'algorithme LMS ajuste les poids synaptiques et le biais du réseau par minimisation de l'erreur quadratique $\varepsilon^2 = (t - o)^2$.

Cette minimisation est conduite en calculant la variation inverse de l'erreur quadratique. Pour un neurone *j*, la variation inverse de l'erreur dû au poids de l'entrée *i* s'exprime par :

$$\frac{\partial \varepsilon_j}{\partial w_{ij}} = -2x_{ij}(o_j - t_j)$$

Le poids est alors modifié d'un pas : $\Delta w_{ij} = -\eta x_{ij}(o_j - t_j)$

Avec η, la constante d'apprentissage.

Le mode incrémental mène généralement plus rapidement à la solution que le mode batch mais est en revanche plus sensible à l'ordre selon lequel sont appliqués les vecteurs d'entrée [Wil-03].

En mode batch, la moyenne de la somme des erreurs quadratiques des N derniers vecteurs présentés en entrée du réseau doit être calculée. Cette moyenne s'exprime à la présentation du vecteur d'entrée k de la manière suivante :

$$\varepsilon_{MOY} = \frac{1}{N}\sum_{k=1}^{N} \varepsilon(k)^2 = \frac{1}{N}\sum_{k=1}^{N} [t(k) - o(k)]^2$$

Il est bien entendu nécessaire d'introduire un critère pour stopper la modification des poids lorsque l'erreur est jugée suffisamment faible. Un seuil θ est introduit et le processus d'apprentissage est stoppé lorsque la condition $\varepsilon_{MOY}<\theta$ est vérifiée. Dans le cas contraire, l'apprentissage se poursuit.
Pour le neurone *j*, chaque poids est modifié en cherchant à minimiser le terme :

$$\frac{\partial \varepsilon_{MOY,j}}{\partial w_{ij}} = -x_{ij}\sum_{k=1}^{N}\left[o_j(k) - t_j(k)\right]$$

C'est-à-dire d'un pas : $\Delta w_{ij} = -\eta x_{ij}\sum_{k=1}^{N}\left[o_j(k) - t_j(k)\right]$

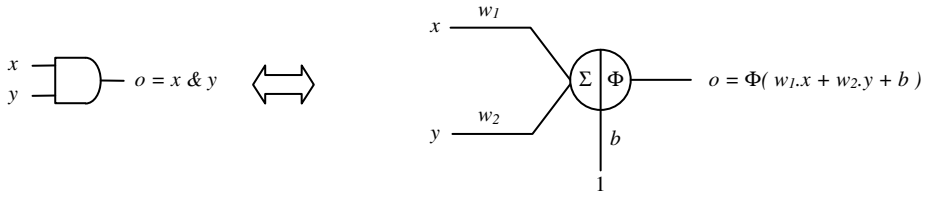
Remarque : Lorsque le neurone est à fonction d'activation non-linéaire, on parle alors de règle d'apprentissage du delta. Dans ce cas, l'expression de la dérivée est explicitement prise en compte dans l'expression du pas de modification des poids synaptiques :

$$\Delta w_{ij} = -\eta x_{ij} \sum_{k=1}^{N} \left[o_j(k) - t_j(k) \right] \frac{df}{dw_{ij}}$$

L'efficacité de l'approche fondée sur la minimisation d'une fonction d'erreur dérivable est très fortement liée à la complexité de la fonction d'activation du neurone. Dans le cas simple d'un réseau à neurone à fonction d'activation linéaire, la mise en œuvre des expressions de correction des poids synaptiques est relativement aisée mais l'utilité demeure cependant limitée pour ce type de réseau (classe des problèmes linéairement séparables). C'est pour cette raison que des réseaux développés notamment sur plusieurs couches faisant par ailleurs appel à des neurones à fonctions d'activation non-linéaire ouvrent la porte aux algorithmes plus complexes tels que la rétropropagation du gradient (*Backpropagation*).

Application de la règle de Widrow et Hoff à l'identification d'une porte AND

Un réseau à un seul neurone est entraîné à mémoriser l'équation logique d'une porte AND à deux entrées à l'aide de la méthode LMS. Le réseau comporte trois entrées x,y et l'entrée unitaire du biais respectivement pondérées par les poids w_1, w_2 et b.



Sur les signaux d'entrée (x,y), les niveaux électriques des états logiques bas et haut sont fixés respectivement par les intervalles [0V;0.4V] et [0.6V;1.0V]1. La mise en place d'une marge de sécurité séparant les deux intervalles permet alors d'introduire davantage de garantie dans le résultat.
Un lot de dix vecteurs d'entrée bruités est appliqué au réseau par la technique du *Batch Learning*. Après initialisation des poids synaptiques, la moyenne de la somme des erreurs quadratiques est calculée en prenant en compte les N derniers exemples présentés au réseau faisant du résultat une moyenne « glissante » quadratique du signal d'erreur. L'apprentissage est stoppé (critère d'arrêt) lorsque la moyenne des N dernières erreurs passe sous un seuil d'acceptance. La figure 2-8 donne l'allure des poids synaptiques au cours de la phase d'apprentissage ainsi qu'une représentation des valeurs retournées en phase de validation. La droite dont les coefficients sont donnés par les valeurs finales des poids synaptiques met en évidence les deux régions pour lesquelles la sortie du réseau répond par un état 0 ou un état 1. L'algorithme LMS ajuste les poids synaptiques et le biais du réseau par minimisation de l'erreur quadratique moyenne. La fonction d'activation du neurone est linéaire, et donc seuls les cas linéairement séparables peuvent être traités. Or la fonction logique AND doit répondre par un niveau logique haut lorsque le couple des signaux d'entrée (x,y) est lui aussi à ce niveau logique et uniquement à ce niveau logique ce qui fait bien de cette fonction un cas qui peut être approché par un réseau à fonction linéaire.

1 Ces intervalles ont été introduits pour apporter davantage de facilité dans la lecture des figures en contribuant à une concentration d'îlots de points.

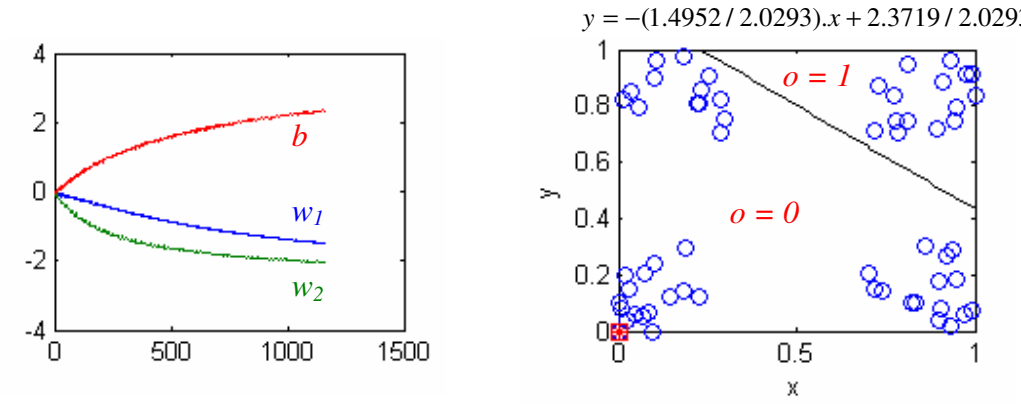


Figure 2-8 Evolution des poids synaptiques au cours de la phase d'apprentissage avec l'algorithme LMS (figure de gauche), et classification d'un jeu de vecteurs de test après apprentissage (figure de droite).

Les valeurs finales des poids synaptiques après arrêt de l'apprentissage sont données dans le tableau 2-1 ainsi que le nombre total de cycles nécessaire au cours du processus.

Algorithme	**Poids synaptiques calculés en fin d'apprentissage**			**Cycles d'apprentissage**
	w_1	w_2	*biais*	
LMS	-1.4952	-2.0293	2.3719	1171

Tableau 2-1 Valeurs des poids synaptiques calculées avec l'algorithme LMS

2.3.3.2 Algorithme Backpropagation (BP) [Ska-96]

L'implémentation de l'algorithme du Backpropagation la plus simple consiste à corriger les poids synaptiques et le biais dans la direction pour laquelle la variation inverse du gradient de l'erreur est la plus forte. Cet algorithme concerne généralement les réseaux multi-couches dont les neurones sont à fonction d'activation $f(x)$ continue et dérivable (classe $C1$) : les fonctions d'activation peuvent donc également s'étendre aux fonctions non linéaires (fonction sigmoïdale par exemple). Cet algorithme est une généralisation de la règle du delta qui se base sur un calcul précis de la dérivée de la fonction d'activation pour calculer la modification des poids synaptiques du réseau. Cependant, si cette méthode est directement applicable pour ajuster les poids de la dernière couche, elle ne l'est pas pour ceux des couches cachées. En effet, on ne connaît pas la sortie désirée pour ces couches, et par conséquent on ne connaît pas directement le terme erreur associé à chaque couche interne. Il est donc nécessaire d'appliquer l'algorithme d'ajustement en partant de la dernière couche et en remontant vers la première ! Sur des réseaux à nombreuses couches, cette rétropropagation de l'erreur introduit des termes à dérivées particulièrement difficiles voir impossibles à implémenter directement dans un circuit analogique.
Considérons l'exemple d'un réseau à L couches représenté sur la figure 2-9.

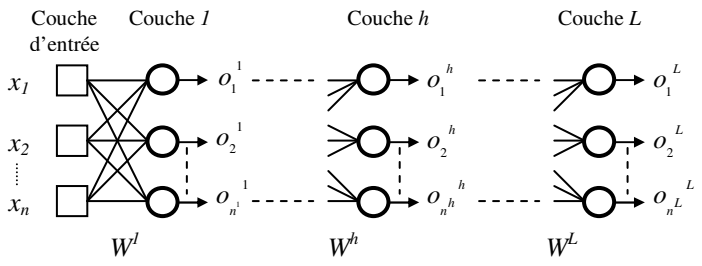


Figure 2-9 Réseau de neurones à L couches

La couche 1 est composée de n^1 neurones, la couche cachée h est composée n^h neurones et la couche de sortie L est composée de n^L neurones.

Le processus d'apprentissage d'un tel réseau doit se poursuivre sous certaines conditions que nous allons présenter.

Définition d'un critère d'arrêt pour l'apprentissage d'un réseau de neurones

L'apprentissage d'un réseau de neurones est typiquement stoppé lorsque l'écart moyen observé entre les sorties de chaque neurone de la couche de sortie et les valeurs attendues passe sous un seuil.

En désignant par o_j^L, la sortie du neurone *j* de la couche L du réseau et par t_j^L, la sortie attendue sur ce neurone, on définit l'erreur PMSE (*Pattern Mean Squared Error*) par :

$$\varepsilon_p = \frac{1}{2}\sum_{j=1}^{n^L}\left(t_j - o_j\right)^2$$

Cette moyenne correspond à la moyenne quadratique de l'écart ($t_j^L - o_j^L$) sur l'ensemble des n^l neurones de sortie de la couche L.

Les algorithmes obéissent à des critères de convergence qui marque l'arrêt de l'apprentissage. Des essais (*cf.* §2.3.3.3) montrent très vite que le PMSE est soumis à des variations qui peuvent être brutales passant par exemple d'une valeur élevée à une valeur extrêmement faible (PMSE<<1). Ceci supposerait alors qu'il y ait, pour un vecteur d'entrée donné, les conditions d'un apprentissage réussi. Il peut être avantageux d'élargir la mesure du PMSE à l'ensemble du lot des vecteurs d'entrée et d'en évaluer la tendance moyenne pour constituer un critère de convergence plus fiable. Il est préférable d'avoir une réponse suffisamment « lissée » et donc globale du réseau qui éviterait toute réaction intempestive et précipiterait l'arrêt de l'apprentissage de manière prématurée. L'erreur quadratique moyenne pour un ensemble de vecteurs d'entrée est alors introduite.
On nomme TPMSE noté ε_T, pour *Total Pattern Mean Square Error*, la fonction de coût du processus d'apprentissage. Le passage de sa valeur sous un seuil fixé est retenu comme critère de convergence pour stopper la phase d'ajustement des poids.
Pour un réseau à n^L sorties et avec un nombre total N d'exemples nécessaires à l'évaluation du TPMSE, ce dernier s'exprime donc par :

$$\varepsilon_T = \frac{1}{2N}\sum_{k=1}^{N}\sum_{j=1}^{n^L}\left(t_j(k) - o_j(k)\right)^2$$

Correction d'un poids synaptique par la méthode de la rétropropagation du gradient :

Le pas de correction du poids entre le neurone j de la couche h et le neurone i de la couche h-1 correspond à (le détail des calculs est donnée en pages annexes de ce chapitre):

$$\Delta w_{ij}^{\ h} = -\eta\frac{\partial \varepsilon_p}{\partial w_{ij}^{\ h}} = -\eta\delta_j^{\ h} o_i^{\ h-1} = -\eta.f'(a_j^{\ h}).\sum_{j=1}^{n^{h+1}}(\delta_j^{\ h+1}.w_{jh}^{\ h+1}).o_i^{\ h-1}$$

Où : - $h \in [1,L]$ est une couche cachée ;
- ε_p est l'erreur quadratique moyenne ;

- η le taux d'apprentissage ;
- δ_j^h la contribution de l'erreur rapportée de la couche h+1 à la couche h.

L'algorithme de la rétropropagation met en œuvre la dérivée de la fonction d'activation de chaque neurone dans la correction des poids synaptiques. L'utilisation d'un tel algorithme suppose par conséquent un certain nombre de conditions :
il est nécessaire d'utiliser une fonction d'activation continue et dérivable c'est à dire de classe C1.
La dérivée de la fonction d'activation doit être explicitable pour être reproduite.
L'application numérique de la dérivée de la fonction d'activation doit restée à une valeur constante au cours temps.

La réunion de ces conditions est possible pour une implémentation de l'algorithme dans un ordinateur ou même un circuit numérique mais demeure techniquement impossible lorsqu'une implémentation sur un circuit analogique est envisagée.

Aussi, l'algorithme qui suit étend son champ d'application aux réseaux de neurones multicouches à fonction d'activation non-linéaire mais non nécessairement continu et dérivable ! De plus, cet algorithme s'adapte aussi à une implémentation analogique.

2.3.3.3 Algorithme Weight Perturbation (WP) [Jab-92]

L'algorithme WP reprend le même principe de l'algorithme BP mais avec la spécificité de ne pas nécessiter la connaissance explicite de la dérivée la fonction d'erreur. La correction des poids se base dans ce cas sur l'estimation de la variation de la fonction d'erreur. La fonction d'activation du neurone peut être non-linéaire et quelconque et pas nécessairement de classe C1.

Son principe se base sur la différence de la mesure du signal d'erreur avant et après une phase dite de perturbation qui consiste à légèrement modifier de façon aléatoire une partie ou l'ensemble des poids du réseau.

En sortie du réseau, la différence du signal d'erreur se mesure par :

$$\Delta\varepsilon_p = \varepsilon_p(w_{ij}^{\ h} + p_{ij}^{\ h}) - \varepsilon_p(w_{ij}^{\ h})$$

Avec :

$\varepsilon_p(w_{ij}^{\ h})$: l'erreur quadratique moyenne avant perturbation du poids *i* du neurone *j* de la couche *h*.
$\varepsilon_p(w_{ij}^{\ h}+p_{ij}^{\ h})$: l'erreur quadratique moyenne après perturbation du poids *i* du neurone *j* de la couche *h*.
$p_{ij}^{\ h}$: la perturbation du poids *i* du neurone *j* de la couche *h*.

L'algorithme se base sur l'approximation au premier ordre du gradient de l'erreur par mesure de la différence $\Delta\varepsilon_p$ selon la variation contrôlée des poids synaptiques. La règle de mise à jour des poids est alors pour la synapse *i* du neurone *j* de la couche *h* :

$$\Delta w_{ij}^{\ h} = -\eta\frac{\partial\varepsilon_p}{\partial w_{ij}^{\ h}} = -\eta\frac{\Delta\varepsilon_p}{\Delta pert(w_{ij}^{\ h})} + O(\Delta pert(w_{ij}^{\ h}))$$

Avec :

$w_{ij}^{\ h}$: le poids synaptique *i* du neurone *j* de la couche *h*.
η : constante d'apprentissage.
$\Delta pert(w_{ij}^{\ h})$: perturbation du poids *i* du neurone *j* de la couche *h*.

$O(\Delta pert(w_{ij}^{h}))$: termes d'ordre supérieurs à deux.

En ne retenant que la variation du premier ordre et en supposant la quantité $\Delta pert(w_{ij}^{h})$ suffisamment petite, l'expression se réduit à :

$$\Delta w_{ij}^{h} \simeq -\eta \frac{\Delta \varepsilon_p}{\Delta pert(w_{ij}^{h})} = -\eta \frac{\varepsilon_p(w_{ij}^{h} + p_{ij}^{h}) - \varepsilon_p(w_{ij}^{h})}{p_{ij}^{h}}$$

Avec :

$\varepsilon_p(w_{ij}^{h})$: l'erreur quadratique moyenne avant perturbation du poids i du neurone j de la couche h.

$\varepsilon_p(w_{ij}^{h}+p_{ij}^{h})$: l'erreur quadratique moyenne après perturbation du poids i du neurone j de la couche h.

p_{ij}^{h} : la perturbation du poids i du neurone j de la couche h.

Le ou les poids perturbés subiront une correction proportionnellement à la différence $\Delta\varepsilon_p$. Si la différence est importante, la correction sera par conséquent elle aussi importante. Si dans le cas contraire, la différence est faible, alors les poids ne sont que très peu affectés et garderont quasiment leur valeur d'origine.

L'algorithme WP nécessite une séquence décomposable en neuf phases que nous présentons maintenant :

Description de l'algorithme :

① Choix d'un vecteur d'entrée à appliquer au réseau.
② Calcul de la sortie du réseau avant perturbation.
③ Calcul du PMSE $\varepsilon_p(w_{ij})$ avant perturbation.
④ Phase de perturbation.
⑤ Calcul de la sortie du réseau après perturbation.
⑥ Calcul du PMSE $\varepsilon_p(w_{ij} + p_{ij})$ après perturbation.
⑦ Calcul de l'incrément du poids Δw_{ij} .
⑧ Mise à jour du poids correspondant.
⑨ Reprendre l'étape 1 tant que l'erreur TPMSE n'est pas inférieure à un seuil donné.

L'algorithme Weight Perturbation peut principalement être implémenté avec deux méthodes de perturbation [Jab-92] : la méthode séquentielle (*Sequential Perturbation*) et parallèle (*Full Perturbation*). La figure 2-10 en donne une illustration. La technique de correction parallèle (*Fully Parallel*) modifie l'ensemble des poids synaptiques du réseau à chaque phase de perturbation, alors que la technique de perturbation dite séquentielle (*Sequential*) excite uniquement un poids par neurone pris au hasard dans l'ensemble du réseau.

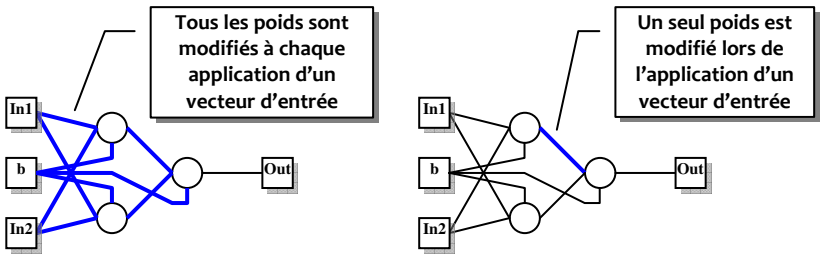


Figure 2-10 Techniques de mise à jour des poids selon la méthode parallèle ou séquentielle.

Application de la méthode WP à l'identification d'une porte AND

L'exemple de mémorisation de l'équation du AND est maintenant appliqué à l'aide de la méthode du Weight Perturbation. Les conditions d'apprentissage sont identiques (présentation des vecteurs d'entrée en *batch learning*). La fonction d'activation du neurone est de type sigmoïdale mais dont la dynamique de sortie sera contenue dans l'intervalle [0,1]. L'expression de la fonction d'activation sera alors de la forme :

$$\Phi(a) = \frac{1+\tanh(a)}{2}$$

Les figures 2-11 et 2-12 donnent l'allure des PMSE, TPMSE, poids synaptiques et du signal de sortie du réseau lorsque les méthodes séquentielle et parallèle sont respectivement employées.

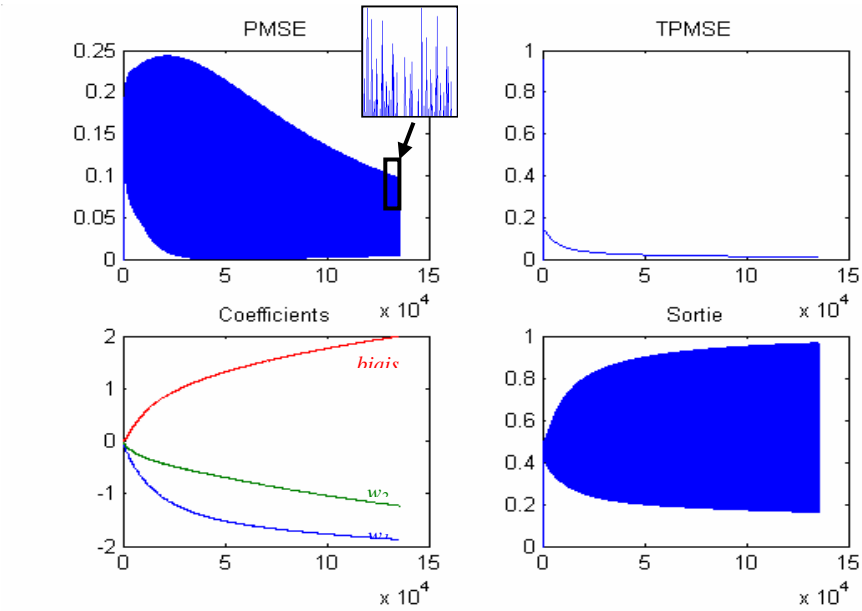


Figure 2-11 Résultats de l'apprentissage Weight Perturbation par la méthode séquentielle

Il apparaît sur ces deux figures, un fort phénomène oscillatoire du signal PMSE. Ceci montre qu'au cours de l'apprentissage, l'erreur en sortie du réseau peut être très variable d'un vecteur à l'autre et que ce phénomène tend à s'atténuer à mesure que les poids du réseau sont corrigés.

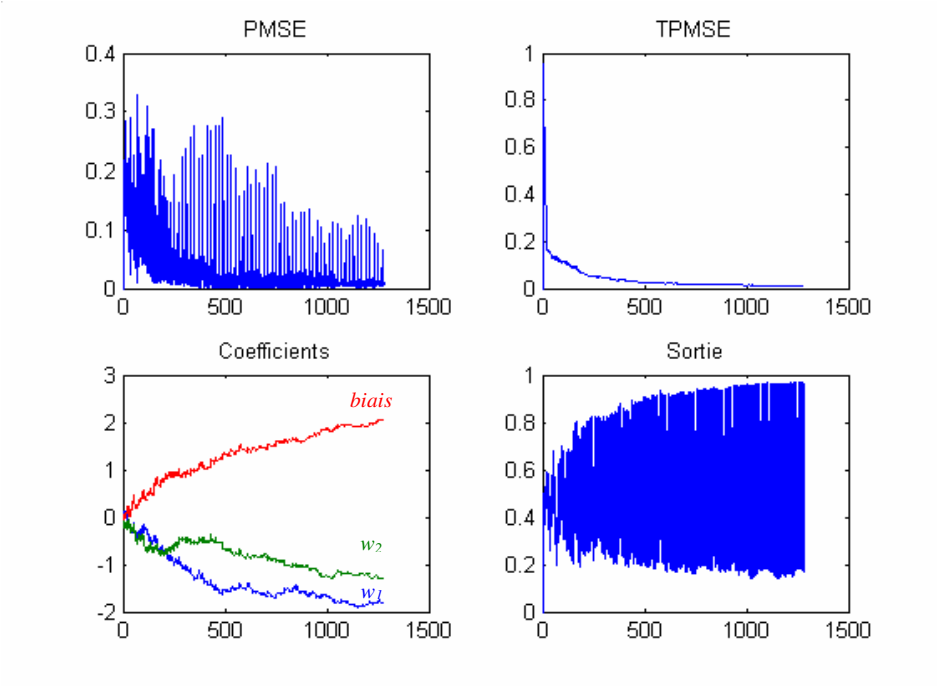


Figure 2-12 Résultats de l'apprentissage Weight Perturbation par la méthode parallèle

Le tableau 2-2 donne les valeurs finales des poids synaptiques obtenus après apprentissage pour les deux méthodes de perturbation séquentielle et parallèle.

Algorithme	**Poids synaptiques calculés en fin d'apprentissage**			**Cycles d'apprentissage**
	w_1	w_2	*biais*	
WP (séquentielle)	-1.8749	-1.2312	1.9972	136002
WP (parallèle)	-1.8202	-1.3106	2.0147	1284

Tableau 2-2 Valeurs des poids synaptiques calculées avec l'algorithme Weight Perturbation (WP) avec la procédure de perturbation séquentielle et parallèle.

On constate tout d'abord que l'algorithme converge pour les deux méthodes de perturbation mais avec des vitesses très différentes : la méthode séquentielle demande près de 100 fois plus de cycles (136002 cycles) que la méthode parallèle (1284 cycles). La méthode séquentielle, comme on pouvait le prévoir, ralentit le processus d'apprentissage de l'algorithme puisque seul un poids est modifié par cycle. Cependant, les valeurs finales des poids calculées sont légèrement différentes entre les deux méthodes. Ceci nous amène à reprendre les fondements d'un tel algorithme d'apprentissage. L'algorithme Weight Perturbation évolue, dans sa phase de perturbation, selon un processus stochastique et, dans le cas de la technique séquentielle, seul un poids (indices *i* et *j* unique par cycle) est modifié dans un sens donné (signe de la perturbation fixé aléatoirement) avec comme conséquence directe une variation du signal d'erreur.

Nous observons d'ailleurs sur la figure 2-11 que les poids synaptiques progressent vers leur valeur finale de façon monotone. Cette progression est le fait d'une modification qui converge vers une valeur finale de façon déterminée (calcul de la dérivée) : la variation du signal d'erreur résulte de la perturbation d'un seul poids et permet une correction ciblée qui va dans le sens d'une diminution du PMSE : le poids responsable de la variation du signal d'erreur est corrigé.

Le mécanisme de perturbation des poids du réseau en mode parallèle est différent puisque tous les poids sont perturbés simultanément à chaque cycle et le sens de modification de ces poids tend à réduire la variation du signal d'erreur. La correction des poids du réseau contribue globalement à une réduction du PMSE. La figure 2-12 montre bien la progression des poids synaptiques du réseau en mode parallèle avec une évolution des poids vers leur position d'équilibre de façon saccadée et pas toujours dans le sens le plus direct. Force est de constater que la vitesse de convergence est bien plus élevée que dans le cas de la méthode séquentielle, et que cela constitue un gain indéniable pour une résolution rapide et efficace de la phase d'apprentissage. On observe par ailleurs que les valeurs finales calculées au terme de l'apprentissage ne sont pas forcément identiques d'une méthode de perturbation à l'autre. Les valeurs initiales données à chacun des poids ainsi que leur modification au cours de l'apprentissage donne cet aspect chaotique dans leur progression. Il en résulte que le chemin pris par chacun des poids est tracé de façon aléatoire et aboutit à une valeur finale qui ne se confond pas forcément avec les valeurs finales observées sous la méthode séquentielle.
Une analyse sur un plus grand nombre d'apprentissage permet néanmoins de se rendre compte que l'on retrouve, en moyenne, des valeurs finales de poids synaptiques plus proches des valeurs obtenues à l'aide de la méthode séquentielle. Cette analyse est illustrée sur la figure 2-13 par le tracé de distributions de type Gaussien caractérisées par une moyenne μ et un écart-type σ. Dans ce cas, les valeurs moyennes qui résultent de cette analyse se rapprochent en effet des valeurs finales obtenues par la méthode séquentielle (tableau 2-3).

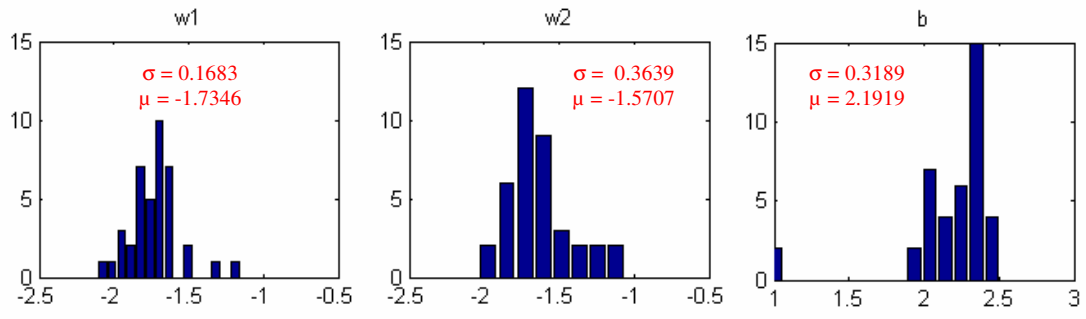


Figure 2-13 Répartition des valeurs des poids synaptiques pour 40 apprentissages convergents (Full Perturbation)

Algorithmes	**Valeurs des poids synaptiques**			**Nombre de cycles**
	w_1	w_2	*biais*	
WP (parallèle)	-1.7346	-1.5707	2.1919	1093
WP (séquentielle)	-1.8749	-1.2312	1.9972	136002
LMS	-1.4952	-2.0293	2.3719	1171

Tableau 2-3 Valeurs moyennes des poids synaptiques calculés avec les algorithmes Weight Perturbation (WP) (méthodes parallèle pour 40 apprentissages convergents et séquentielle) et LMS

La figure 2-14 rend bien compte de la manière dont l'algorithme ajuste ses coefficients pour traiter le problème c'est-à-dire de trouver une droite qui sépare au mieux les couples d'entrée (x,y). Un couple (x,y) au-delà de cette droite donne un niveau haut au neurone de sortie et un niveau bas dans les cas restants (x ou y au niveau bas).

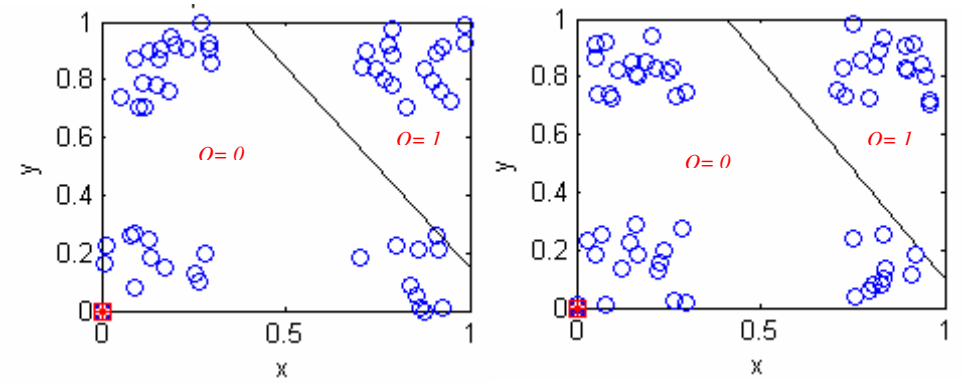


Figure 2-14 Réponses du neurone (cercles) avec droite de séparation par l'algorithme Weight Perturbation (méthodes parallèle à gauche et séquentielle à droite).

Le tableau 2-3 donne des résultats très comparables qui démontrent l'efficacité avec laquelle le processus d'apprentissage du Weight Perturbation (par la technique séquentielle) évolue de manière approchée à celui du LMS. Dans l'apprentissage par Weight Perturbation séquentiel la correction est effectuée sur un seul coefficient synaptique par cycle d'apprentissage ce qui signifie que seule une variation dans un sens précis de ce coefficient est responsable d'une diminution du signal d'erreur. Dans l'apprentissage par LMS, la correction est réalisée sur l'ensemble des poids synaptiques par cycle d'apprentissage. Dans les deux cas (l'algorithme LMS et Weight Perturbation), la correction des poids s'effectue dans le bon sens et les valeurs des poids synaptiques évoluent selon une direction optimale. La convergence est « linéaire » pour les deux techniques d'apprentissage avec une évolution des poids synaptique comparable.

2.3.4 Choix de l'apprentissage

La dualité architecture – algorithme d'apprentissage nous amène à considérer, dans le cas d'un réseau à architecture non bouclée, les algorithmes limités à la classe des apprentissages supervisés dont nous retiendrons les plus communs à savoir l'algorithme de Widrow et Hoff (LMS), l'algorithme Backpropagation et l'algorithme Weight Perturbation. Si la difficulté d'implantation dans un circuit d'un réseau repose avant tout sur l'architecture même d'un neurone dont les solutions techniques sont à ce jour connues et réalisables, cela est moins évident lorsqu'il s'agit d'implémenter l'algorithme d'apprentissage sous la forme d'un circuit à composants analogiques. En effet, les règles d'apprentissage obéissent à des lois mathématiques dont, pour certaines, la reproduction physique sur un circuit électronique est au mieux imprécise ou au pire tout simplement techniquement impossible à réaliser. Au mieux, on parvient à rapprocher la loi par une batterie de fonctions de l'électronique qui peuvent très vite atteindre un niveau de complexité tel que les critères de conception dépassent les limites du cadre fixés par les enjeux actuels (consommation, coût, …) et cette solution n'est donc pas viable. Dans les circuits numériques, avec les outils puissants dont dispose tout concepteur actuellement, il est possible de reproduire une fonction avec un minimum de lignes de code et un certain nombre d'applications fonctionnant à partir d'un réseau de neurones ont déjà été programmées dans des FPGA. Cette implémentation dans des circuits numériques est réalisable mais au prix d'une consommation énergétique et en surface supérieure à une implémentation analogique. L'implémentation du réseau dans un circuit analogique nécessite prudence et réflexion. Pour cela, un regard doit être porté sur la façon dont est exécuté l'algorithme et la règle mathématique utilisée pour corriger les poids synaptiques du réseau.

L'algorithme de Widrow et Hoff met en application un réseau dont les neurones sont à fonction d'activation linéaire. Les sorties sont par conséquent une combinaison linéaire de leurs entrées. Cette méthode d'apprentissage trouve un terrain d'application assez restreint en limitant ses traitements à des classes de problèmes de classification séparables linéairement. Pour s'approcher des comportements naturels qui nous entourent et élargir le champ applicatif, la fonction de sortie du neurone doit être de préférence non linéaire et l'algorithme de rétropropagation du gradient ou *Back Propagation* porpose des règles de modification des poids pour ce type de neurone.

Pour un processus d'apprentissage supervisé, l'architecture classique du réseau de neurones (feedforward) est basée sur l'algorithme *Back Propagation*. Cette architecture met en œuvre un algorithme de minimisation de fonction de coût par descente de gradient. En ce sens, il convient de calculer le gradient de la fonction de coût pour chaque exemple d'apprentissage. Ce calcul est lourd tant en architecture électronique qu'en temps de calcul numérique. Une alternative attractive est l'algorithme *Weight Perturbation* qui se base sur l'estimation du gradient de la fonction de coût [Jab-92, Val-02]. Un tel algorithme est implantable sous forme analogique concentrant dans un circuit un processus de calcul « massivement parallèle ». Ce bénéfice est néanmoins contre balancé par la précision de l'estimation dont la conséquence principale est la vitesse d'apprentissage. Il sera alors important d'analyser l'influence des limitations dues à l'implémentation analogique sur la vitesse d'apprentissage.

L'algorithme *Weight Perturbation* base la correction des poids synaptiques non pas sur un calcul précis de la variation des poids mais sur son estimation [Jab-92, Val-02, Mic-04]. De fait, l'expression nécessaire au calcul des poids est réduite à la différence des erreurs en sortie du réseau provoquée à la suite d'une phase dite de perturbation : un ou l'ensemble des poids est légèrement écarté de sa valeur courante entraînant une variation du signal en sortie du réseau par rapport à la valeur du signal de référence (valeur cible). En réponse, le poids synaptique est modifié d'un pas pour réduire cette variation. L'apprentissage est stoppé (critère d'arrêt) au passage de la moyenne des N dernières erreurs quadratiques moyennes (TPMSE) sous un seuil de validation de l'apprentissage.

La mise en œuvre de cet algorithme et en particulier de son implémentation dans un circuit analogique est pour nous essentielle. L'algorithme *Weight Perturbation* sera par conséquent le candidat retenu et nous aborderons dans le prochain chapitre l'aspect modélisation de l'ensemble réseau-algorithme.

2.4 Conclusion

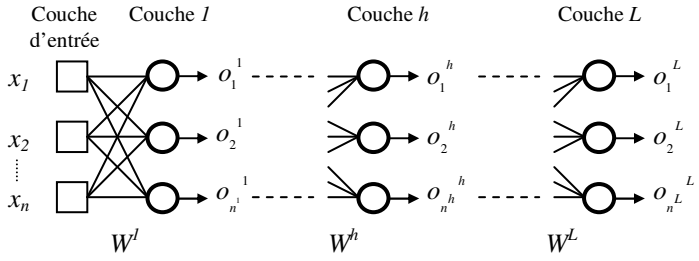
Nous avons présenté les principales familles de réseaux de neurones et les méthodes d'apprentissage associées. Ce travail constitue notre première étape dans la conception analogique d'une architecture à traitement massivement parallèle. Cette étape nous a permis d'évaluer la faisabilité de l'architecture neuronale et de son algorithme d'apprentissage sous un environnement très général fixé par l'outil MATLAB. Nous avons en effet sélectionné un réseau de neurones non bouclé (feedforward) capable de se spécialiser dans des opérations de classification à prise de décision rapide. Nous avons également fait le choix d'une méthode d'apprentissage et de l'algorithme *Weight Perturbation* qui présente des qualités que l'on ne peut pas ignorer en conception analogique. Cet algorithme se limite en effet à mesurer le sens de variation du signal d'erreur en sortie du dispositif d'apprentissage pour corriger la valeur des poids synaptiques. Cette caractéristique rend possible l'implémentation matérielle du réseau analogique et de son algorithme d'apprentissage.

On utilisera par conséquent le réseau feedforward et son algorithme d'apprentissage Weight Pertubation en mode de perturbation parallèle pour aborder un problème de classification plus complexe qu'est la reconnaissance de caractères. La mise en œuvre du réseau de neurones et de son apprentissage fait intervenir des composants analogiques non exempts de défauts. Nous espérons, à travers cette application, démontrer que la boucle de rétroaction en phase d'apprentissage permettra de compenser les défauts intrinsèques de ces composants.

2.5 Annexe du chapitre 2

Définition des expressions nécessaires à la correction des poids synaptiques d'un réseau multicouches à l'aide de la méthode par rétropropagation.

Considérons ci-dessous le réseau à L couches.



La couche *1* est composée de n^1 neurones, la couche h est composée n^h neurones et la couche L est composée de n^L neurones.

Le pas de correction du poids entre le neurone j de la couche h et le neurone i de la couche *h-1* correspond à :

$$\Delta w_{ij}^{\ h} = -\eta \frac{\partial \varepsilon_p}{\partial w_{ij}^{\ h}}$$

Où : $h \in [1,L]$, ε_p est l'erreur quadratique moyenne et η le taux d'apprentissage.

En introduisant le potentiel d'activation $a_j^{\ h}$ du neurone j de la couche h, il vient :

$$\Delta w_{ij}^{\ h} = -\eta \frac{\partial \varepsilon_p}{\partial a_j^{\ h}} \frac{\partial a_j^{\ h}}{\partial w_{ij}^{\ h}}$$

La variation de l'erreur quadratique moyenne en sortie du neurone j de la dernière couche L s'exprime par :

$$\delta_j^{\ L} = -\frac{\partial \varepsilon_p}{\partial a_j^{\ L}} = -\frac{\partial o_j^{\ L}}{\partial a_j^{\ L}}(o_j^{\ L} - t_j^{\ L}) = -f^{'}(a_j^{\ L}).(o_j^{\ L} - t_j^{\ L})$$

Où le terme $f^{'}(a_j^{\ L})$ correspond à la dérivée de la fonction d'activation.

Il en va de même pour la couche cachée h :

$$\delta_j^{\ h} = -\frac{\partial \varepsilon_p}{\partial a_j^{\ h}} = -f^{'}(a_j^{\ h}).\sum_{j=1}^{n^{h+1}}(\delta_j^{\ h+1}.w_j^{\ h+1})$$

Où le terme $\sum_{j=1}^{n^{h+1}} (\delta_j^{h+1}.w_j^{h+1})$ correspond à la contribution de l'erreur de la couche *h+1* et de tous ses neurones rétropropagée sur la sortie du neurone *j* de la couche *h*. Cette rétropropagation est poursuivie jusqu'à atteindre le première couche où le terme δ_j^1 sera à son tour calculé.

$$\frac{\partial a_j^h}{\partial w_{ij}^h} = \frac{\partial}{\partial w_{ij}^h}\left(\sum_{i=1}^{n^{h-1}} w_{ij}^h o_i^{h-1}\right) = o_i^{h-1}$$

Par conséquent, le pas de correction du poids entre le neurone *j* de la couche *h* et le neurone *i* de la couche *h-1* s'écrit finalement :

$$\Delta w_{ij}^h = -\eta \delta_j^h o_i^{h-1}$$

Chapitre 3 - description en VHDL-AMS d'un réseau de neurones analogique dédié à la reconnaissance de caractères

3.1 Introduction

Le chapitre précédent nous a permis d'évaluer la faisabilité d'un réseau de neurones pour traiter des problèmes de classification. Ce travail s'inscrit dans le cadre de la conception de systèmes à traitement massivement parallèle où les réseaux de neurones offrent des réponses particulièrement adaptées. Une architecture de réseau de type *feedforward* a été retenue à laquelle nous avons associé l'algorithme Weight Perturbation. Cette étude prépare la réalisation d'une application spécifiquement adaptée à traiter des problèmes de reconnaissance de caractères.

Dans la chronologie des étapes de conception, si la faisabilité du principe est établie, on passe à l'étape du partitionnement puis de la sélection d'une architecture. C'est cette étape que nous abordons dans ce chapitre. La description de chacune des fonctions devra d'une part prendre en compte les comportements les plus significatifs autour d'un modèle comportemental et d'autre part introduire progressivement l'influence de la technologie des composants à l'aide d'un modèle de plus bas niveau d'abstraction. La progression conduira à la description d'un modèle structurel de premier niveau.

Le développement de la solution technique permettant d'initier un premier lien avec les simulations électriques est abordé selon une approche descendante (Top-Down). Cette approche permet de translater les spécifications système vers les spécifications circuits tout en assurant la remontée des limitations électriques et technologiques vers les cahiers des charges. L'interaction entre les différentes fonctions est à présent assurée par des connexions électriques. Par conséquent, le développement de l'architecture doit se faire dans un environnement unifié, supportant l'ensemble de la hiérarchie de conception jusqu'à la phase de vérification (et donc compatible avec SPICE). Le langage VHDL-AMS est le candidat que nous avons retenu pour cela. Il permettra de décrire le système complet du réseau de neurones avec ses blocs électroniques (modèles comportementaux et structurels) et les algorithmes pour le processus d'apprentissage.

Dans ce contexte de translation, les modèles comportementaux du réseau et de son algorithme d'apprentissage décrits sous MATLAB sont d'abord transposés vers l'environnement unifié de SMASH qui supporte le langage VHDL-AMS. Cette transposition est indispensable et permet de démontrer l'équivalence de ces modèles. Elle donne la garantie que le langage VHDL-AMS permet d'assurer sans erreur le passage des données à travers tous les niveaux hiérarchiques de l'application.

Ce chapitre répond finalement à trois objectifs qui seront présentés distinctement en trois parties. La première partie fera la présentation du réseau de neurones qui sera entraîné à reconnaître des caractères alphanumériques. Une étude de faisabilité du réseau et de son apprentissage conduite sous MATLAB sera proposée. La seconde partie permettra de s'assurer de la fiabilité de la description en VHDL-AMS du réseau et de son algorithme d'apprentissage. Les modèles comportementaux du réseau et de son algorithme d'apprentissage décrit sous MATLAB seront transposés en VHDL-AMS puis comparés. Dans la troisième partie, une architecture fonctionnelle du réseau de neurones sera proposée en VHDL-AMS. La description de cette architecture sera progressivement affinée et permettra de mesurer l'influence des choix architecturaux sur le comportement global du réseau. L'architecture sera développée autour d'une branche synaptique du réseau dont le choix de la topologie suffira à poser les bases pour la phase de dimensionnement.

3.2 Présentation de l'application dédiée à la reconnaissance de caractères

Le domaine de la reconnaissance automatique de caractères bénéficie par son côté fascinant d'une certaine popularité. Il met en œuvre un processus complexe de classification non explicitable au sens mathématique pour lequel les solutions techniques à base de réseau de neurones sont particulièrement efficaces.
L'application retenue dans cette thèse est la reconnaissance de caractères. L'application sera fortement simplifiée afin de se focaliser sur les points critiques de la conception d'un tel système. Nous avons choisi pour cela d'entraîner un réseau de neurones à classer des caractères numériques. Nous limiterons la description à la reconnaissance d'un seul chiffre (0 à 9). La réalisation pourra ensuite être généralisée à un nombre plus important de neurones c'est-à-dire d'augmenter la capacité du réseau à reconnaître davantage de chiffres, de nombres ou pourquoi pas de caractères alphabétiques.

La représentation symbolique de l'ensemble de ces chiffres est donnée sur la figure 3-1.

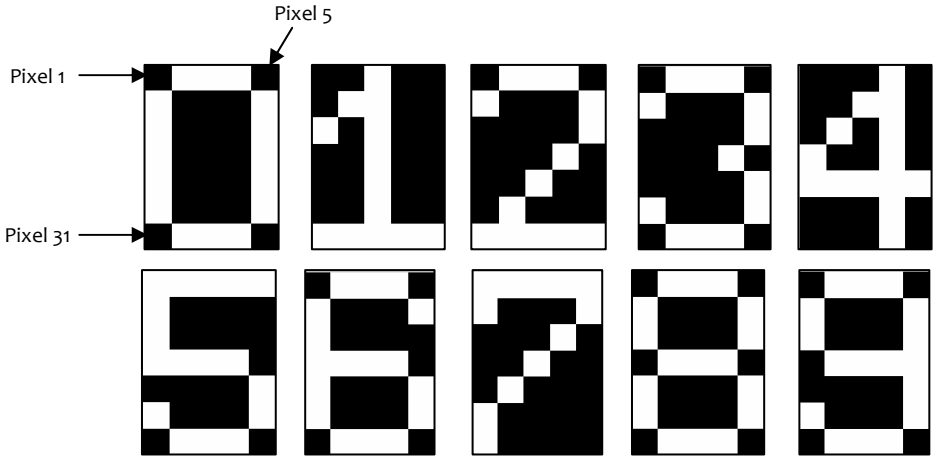


Figure 3-1 La représentation symbolique de l'ensemble des chiffres en niveaux noir et blanc

Le cliché de la figure 3-2 montre toute la difficulté qu'aura le réseau à reconnaître des chiffres particulièrement déformés par l'ajout d'un bruit. Cette déformation conduit très vite à une erreur d'interprétation entre chiffres : le chiffre 3 peut être confondu avec le chiffre 0, le chiffre 5 ou encore avec le chiffre 8 ou 9, etc..... Le bruit est ici traduit en un niveau noir et blanc et repose sur le principe de seuillage.

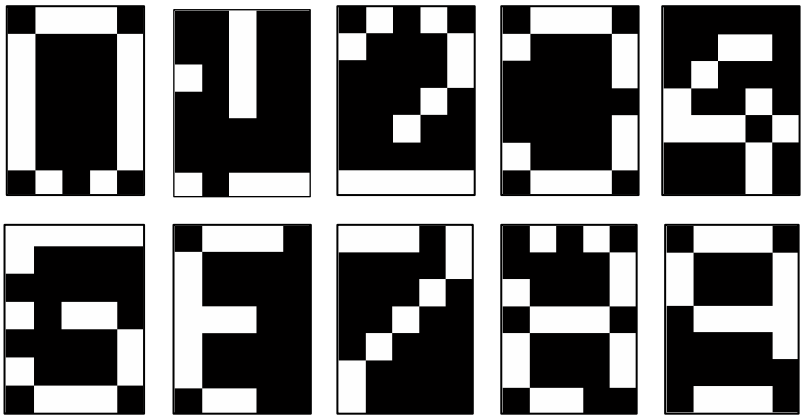


Figure 3-2 Représentation des chiffres bruités après opération de seuillage des niveaux de gris en niveaux noir et blanc

3.2.1 Principe de génération d'une image bruitée

Le vecteur d'entrée présenté au réseau consiste à reproduire une matrice représentative de l'état d'excitation de chaque pixel. L'image originale se traduit par un niveau d'éclairement du pixel gradué en niveaux de gris. En introduisant un étage de pré traitement, nous accentuons les erreurs en traduisant les niveaux de gris en niveaux noirs ou blancs par une opération de seuillage. Cette opération permettra de rendre le niveau de bruit d'un pixel purement binaire et laissera au réseau une plus grande marge de tolérance sur la valeur à attribuer aux poids[2]. La figure 3-3 donne un exemple de seuillage de la matrice de pixels bruitée du chiffre 1. La couleur noire est attribuée au pixel si ce dernier présente un niveau compris dans l'intervalle [0.0,*FullScale*/2[et blanc dans l'intervalle [*Fullscale*/2,*FullScale*]. En considérant par exemple la valeur de pleine échelle égal à 1.0, il vient que le pixel sera de couleur noire si son niveau est compris dans l'intervalle [0.0,0.5[et blanc dans l'intervalle [0.5,1.0]. Les couleurs sont donc inversées lorsque les niveaux de bruit sont importants et donnent une représentation déformée du chiffre[3].

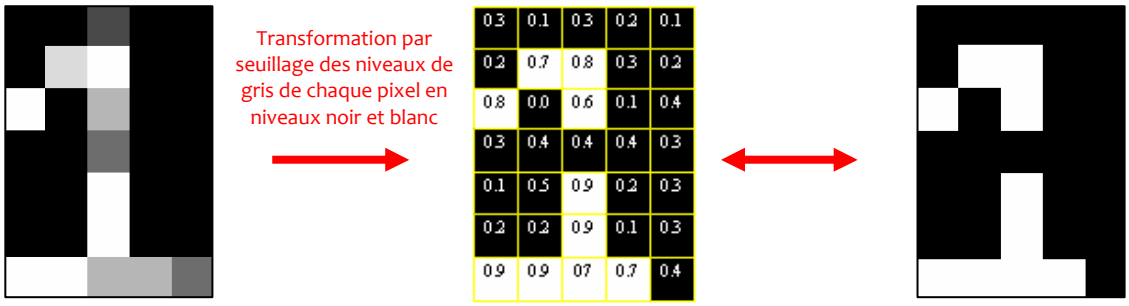


Figure 3-3 Représentation matricielle du chiffre 1 bruité.

3.2.2 Principe de la reconnaissance de caractères par réseau de neurones

Dans notre étude, nous nous focaliserons davantage sur l'aspect technique et donc structurel d'un neurone que sur la performance du réseau à traiter des images à haute résolution. Aussi, nous limiterons la dimension de représentation d'un chiffre à une matrice de 7 pixels de hauteur sur 5 pixels de largeur.

Sous sa forme la plus simple, le réseau de neurones comporte dix neurones (N_0 à N_9) pour reconnaître les dix chiffres (figure 3-4). Chaque neurone reçoit le signal des 35 pixels plus son propre biais (b_0 à b_9). Chaque neurone doit alors prendre la décision binaire de reconnaissance du caractère qu'il doit reconnaître. Ainsi sa sortie sera 0 s'il ne reconnaît pas le chiffre et à 1 dans le cas contraire.

2 Si on travaille avec des niveaux de gris uniquement, il est alors plus facile de reconnaître les caractères mais les valeurs des poids devront être plus précises.
3 La représentation des niveaux a été ramenée à une décimale pour des raisons de clarté

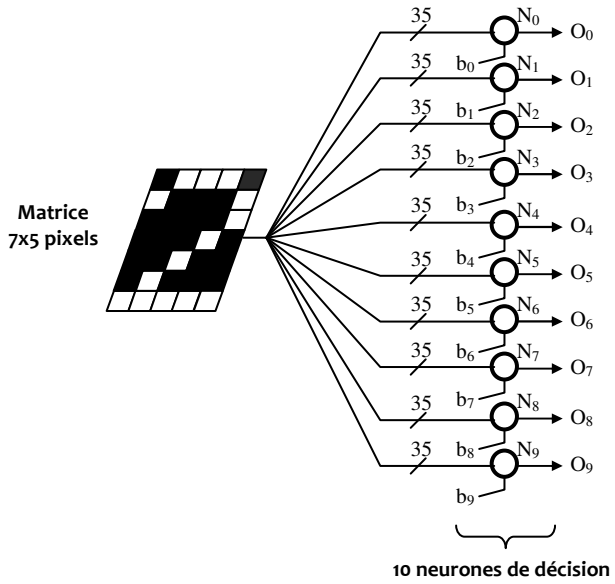


Figure 3-4 Dispositif de reconnaissance des chiffres 0 à 9.

Le dispositif de reconnaissance est illustré partiellement sur la figure 3-5 et se limite pour des raisons de clarté à deux neurones de sortie. Chaque neurone est ici supposé reconnaître le chiffre 1 ou le chiffre 2.

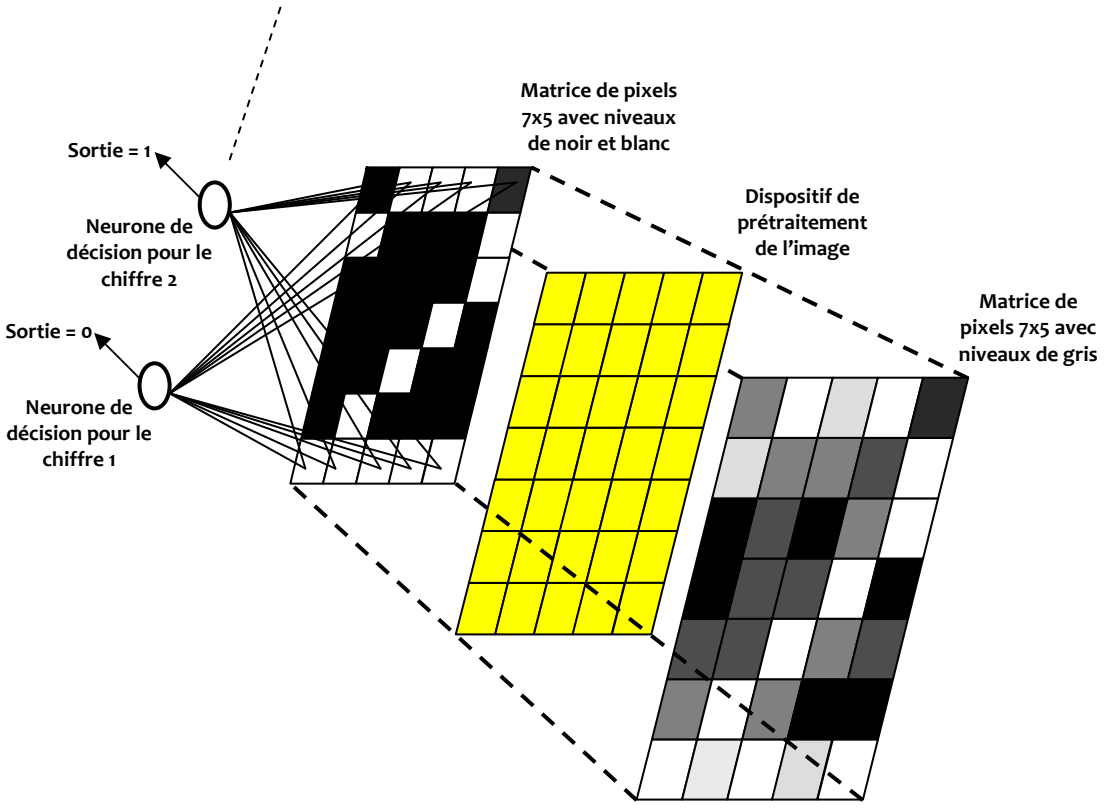


Figure 3-5 Dispositif de reconnaissance limité aux chiffres 1 et 2.

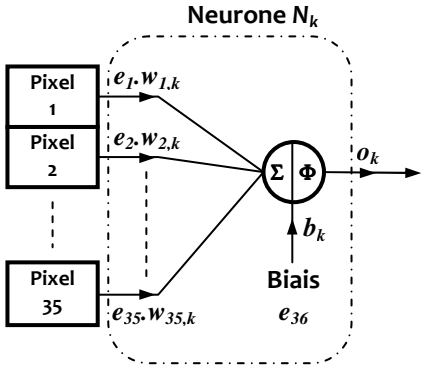
Chaque neurone du réseau est à fonction d'activation tangente hyperbolique garantissant les propriétés de continuité et de dérivabilité. La nécessité de telles propriétés découle de l'utilisation de l'algorithme Backpropagation et en ce qui nous concerne de l'utilisation de l'algorithme Weight perturbation. La fonction tangente hyperbolique formate la dynamique du signal de sortie dans l'intervalle [0,1]. La présence d'un seuil fixé à la valeur 0,5 permet dans ce cas d'apporter une décision binaire sur l'intervalle [0.0 , 0.5[∪ [0.5 , 1.0].

Pour simplifier les explications, toutes les grandeurs ont été normalisées à l'intervalle [0,1]. Ainsi, la sortie en tension du neurone sera arbitrairement fixée à [0V,1V] mais pourra être dénormalisée à [V_{SS}, V_{CC}] avec $V_{SS} < V_{CC}$.

Nous allons maintenant présenter le réseau de neurones.

3.2.3 Le réseau de neurones

Comme chaque neurone est entraîné à la reconnaissance d'un caractère, nous nous focaliserons sur un seul neurone. Notre étude portera spécifiquement sur un réseau entraîné à reconnaître un unique caractère et plus particulièrement l'un des dix chiffres de la base de numération décimale. La figure 3-6 donne le schéma d'interconnexion entre les pixels de la matrice et les entrées du neurone N_k. Chaque pixel étant relié à une entrée distincte du neurone, le réseau sera donc constitué de 35 entrées et d'une entrée supplémentaire de biais.



$$o_k = \phi(p) = \phi\left(\frac{1+th(p)}{2}\right)$$

on pose généralement : $p = e_{36}.b_k + \sum_{i=1}^{35} e_i.w_i$

qui devient avec : $e_{36} = 1 : p = b_k + \sum_{i=1}^{35} e_i.w_i$

Figure 3-6 Réseau à reconnaissance d'un caractère.

Le réseau est, dans sa phase d'apprentissage, entraîné par l'algorithme de correction de poids *Weight Perturbation* dont l'implémentation dans du silicium a été proposée dans [Dio-00]. La conception d'un modèle neuronal ne se réduit pas au choix du nombre de neurones dans la couche cachée du réseau et à la bonne exécution d'un algorithme d'apprentissage mais également à la qualité de présentation de ses vecteurs d'entrée. En effet, les vecteurs sont présentés à la manière du « *Batch Learning* » [Dio-00, p2.4]. Le vecteur représentatif du chiffre à apprendre au réseau est présenté alternativement avec un autre vecteur représentant un chiffre pris aléatoirement entre 0 et 9. On a donc une série de vecteurs dont un échantillon sur deux est le référent4. Cette méthode de présentation des vecteurs doit pouvoir donner au réseau une meilleure faculté de discernement du chiffre à reconnaître et lui permettre d'acquérir une certaine représentation de la population qu'il sera amené à différencier une fois entraîné. Un

4 Cette méthode de représentation est spécifique à notre étude puisqu'un seul neurone est ici entraîné. Si la méthode doit être étendue à l'apprentissage des dix neurones du réseau, la méthode devient alors plus complexe.

autre critère important est de rendre le réseau apte à traiter et reconnaître des vecteurs « pollués » c'est à dire entachés de bruit. Ce critère doit donc être pris en compte au moment de l'apprentissage en ajoutant une composante de bruit à chacun des vecteurs. Néanmoins, l'ajout de cette composante doit être contrôlé de sorte que le réseau ne s'ajuste pas trop spécifiquement au bruit présent dans les données d'apprentissage. Lors de la phase de validation, le bruit présent dans ces vecteurs, pouvant différer de celui de la base d'apprentissage, induira trop d'erreur. En conséquence, les erreurs augmenteront en phase de fonctionnement (hors apprentissage) lorsque le bruit n'aura pas exactement la même caractéristique statistique que celle dans la base d'apprentissage. Ceci limite donc la généralisation de la prise de décision à des entrées bruitées ! En d'autres termes, il faut trouver un modèle qui réalise le meilleur compromis entre les capacités d'apprentissage et les capacités de généralisation : si le réseau apprend « trop bien », il s'ajuste aux particularités des vecteurs de la base d'apprentissage (au bruit de ces vecteurs), et donc a de mauvaises performances de généralisation [Dre-04, p103]. Ce compromis a été formalisé sous le terme de « dilemme biais/variance » [Gem-92]. Un modèle, possédant un grand nombre de paramètres ajustables, peut donc avoir un biais très faible, c'est à dire s'ajuster aux données quelque soit le bruit présent dans celles-ci, mais il risque d'avoir une variance très grande (erreur importante), c'est à dire être très dépendant de la réalisation particulière du bruit présent dans les données d'apprentissage : le modèle est surajusté. En revanche, un modèle, possédant peu de paramètres ajustables, peut être très dépendant du bruit présent dans les données c'est à dire très sensible à l'ensemble des données utilisée, mais s'avérer incapable d'être proche de la régression (erreur constante mais pas forcément élevée) : modèle trop généraliste.
Pour contrôler les performances de l'algorithme d'apprentissage, l'erreur PMSE5 doit servir d'indicateur en considérant les valeurs prises sur un lot de vecteurs d'entrée. Ainsi, en moyennant ces valeurs, on obtient une réponse plus « lissée » et moins assujettie au bruit qui atténue le phénomène de surajustement et réalise une forme de régularisation. Cette réponse s'exprime par le terme de *Total Pattern Mean Square Error* ou TPMSE (*cf.* §2.3.3.2) et constituera la fonction de coût à minimiser. Ce processus de minimisation aura pour critère d'arrêt le passage sous un seuil : l'algorithme sera stoppé et la phase d'apprentissage du réseau réalisée.

3.3 Etude de faisabilité

Nous abordons l'étude de faisabilité de notre application. Celle-ci doit déterminer les spécifications générales du système à un très haut niveau d'abstraction. On doit donc disposer d'un simulateur fonctionnel et Matlab est généralement l'outil le plus utilisé à ce stade du flot de conception.

3.3.1 Description de l'application sous MATLAB

Dans notre application, la phase d'apprentissage du réseau à décrire s'appuie sur une phase préparatoire. Cette phase préparatoire consiste à générer d'une part les vecteurs cibles et d'autre part les vecteurs d'entrée bruités par une composante uniformément. Ces vecteurs sont stockés dans deux matrices distinctes : la matrice des vecteurs d'apprentissage et la matrice des vecteurs cibles. Cette dernière contient les valeurs attendues en sortie du réseau pour chaque vecteur d'entrée. Ces valeurs sont définies en considérant la réponse d'un neurone normalisée à l'intervalle [0,1]. Nous choisirons deux valeurs nominales prises respectivement à 10% et 90% de la valeur pleine échelle c'est à dire {0.1, 0.9} afin de prendre en compte la contribution du bruit.

5 Par définition, l'erreur PMSE associe pour un motif le calcul de l'erreur moyenne sur *k* neurones de sortie. Dans notre cas, le calcul de l'erreur revient à considérer qu'une seule sortie d'un neurone et le terme de Pattern Error serait plus adapté.

La figure 3-7 offre le synoptique des différentes phases qui seront décrites sous MATLAB. Notons que la phase d'apprentissage sera stoppée en fonction de critères que nous détaillerons plus loin.

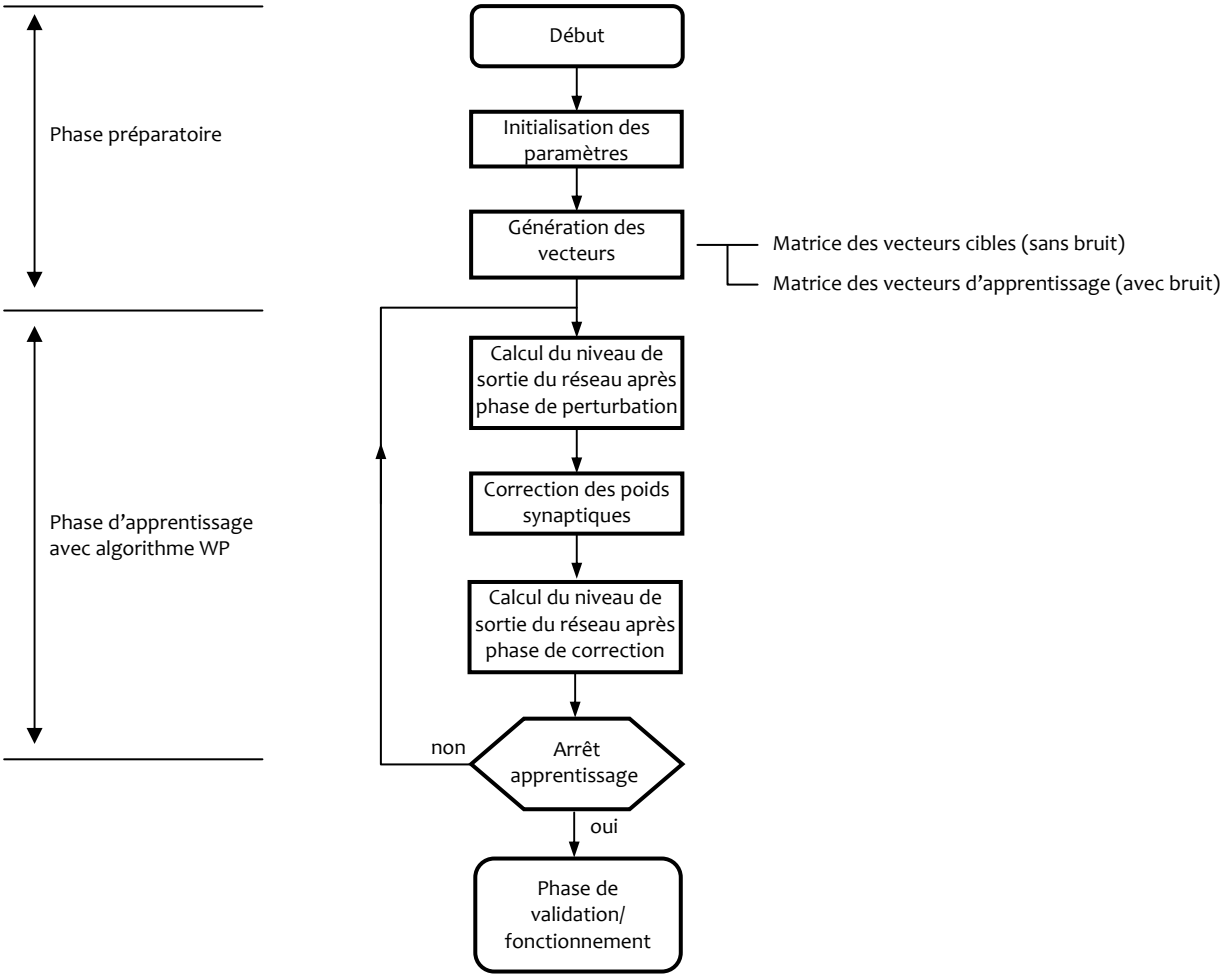


Figure 3-7 Synoptique de la description sous MATLAB

3.3.2 Analyse de faisabilité

Le réseau est maintenant entraîné à reconnaître le chiffre 1 par la méthode du *Full Perturbation*. L'évolution du TPMSE ainsi que les poids au cours d'un apprentissage sont représentés sur la figure 3-8. L'algorithme d'apprentissage est stoppé au passage du TPMSE sous le seuil d'acceptance après 264 cycles.

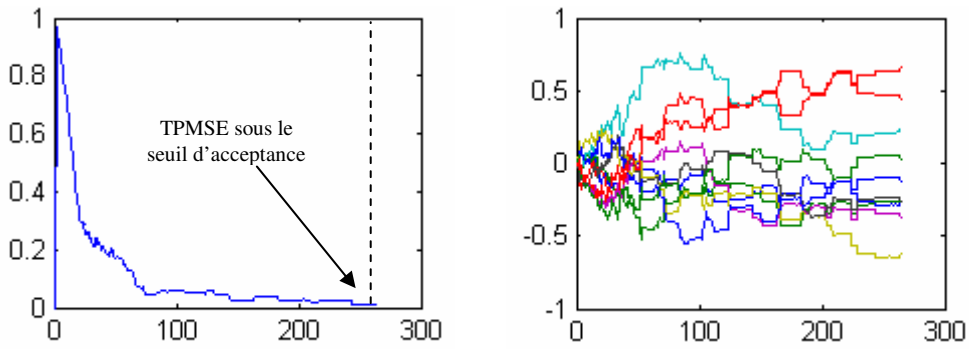


Figure 3-8 Evolution au cours de l'apprentissage du TPMSE (figure de gauche) et des poids synaptiques du réseau (figure de droite). Seuls les dix premiers poids sont représentés ici pour en faciliter la lisibilité.

Le réseau paramétré est ensuite testé en phase de validation. Il s'agit de soumettre au réseau une série de 30 vecteurs (chiffres de 0 à 9) dont un vecteur sur deux correspond au chiffre que le réseau a été entraîné à reconnaître (chiffre 1). A ces vecteurs une composante de bruit de type gaussien peut être ajoutée autour des valeurs nominales {0.1,0.9} avec un écart-type σ. Le tableau 3-1 donne la répartition des réponses du réseau lorsque le bruit gaussien est progressivement ajouté aux vecteurs d'entrée. De façon à augmenter le taux de bonne classification, une marge de sécurité est ajoutée. La réponse du neurone sera correcte si :

- la réponse du neurone est incluse dans l'intervalle [0.0,0.4] en cas de présentation d'un chiffre différent de 1 ;
- la réponse du neurone est incluse dans l'intervalle [0.6,1.0] en cas de présentation du chiffre 1.

Si la réponse du neurone appartient à l'intervalle]0.4,0.6[, aucune décision ne peut être prise.

Lorsque les vecteurs sont transmis au réseau sans ajout de bruit, la reconnaissance du chiffre 1 a été effectuée sans la moindre erreur. En ajoutant un bruit gaussien d'une amplitude modérée (écart-type $\sigma=(0.2)^{1/2}$), le taux de décisions prises correctement sur les 30 vecteurs est proche de 97% avec une seule réponse non classée c'est-à-dire appartenant à l'intervalle]0.4,0.6[. Dans le cas extrême où les vecteurs d'entrée sont fortement bruités (écart-type $\sigma=(0.4)^{1/2}$), le réseau a produit trois réponses qui se situent dans l'intervalle]0.4,0.6[et pour lesquelles aucune décision ne peut être prise. Cette situation démontre qu'un niveau décisionnel supplémentaire avec de nouveaux critères permettrait, en ajoutant une couche de neurones en sortie, de rejeter de façon sûre les réponses incertaines.

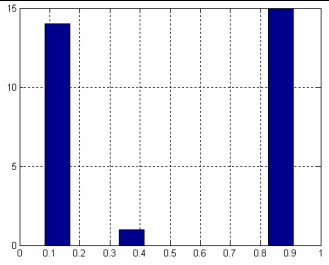
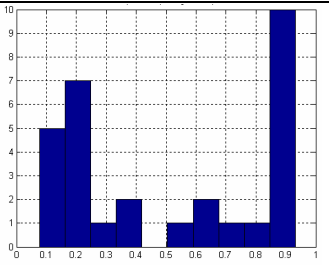
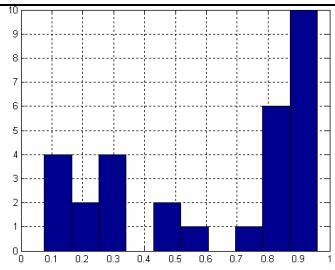
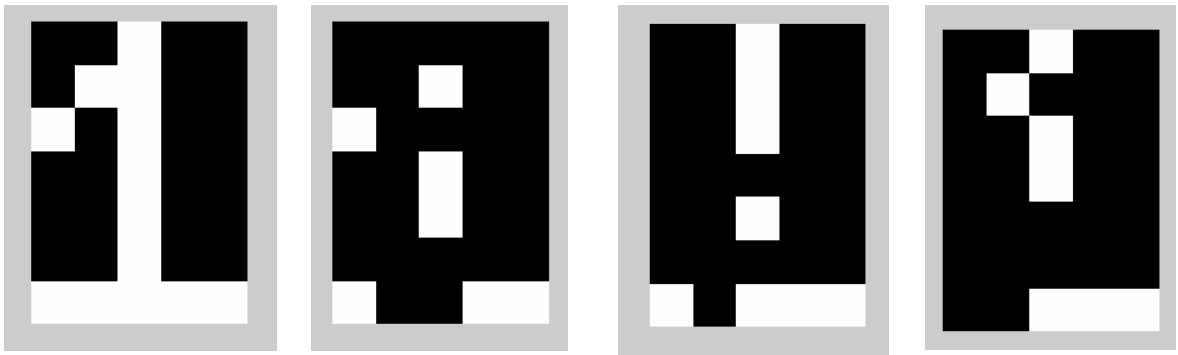
	Vecteurs d'entrée sans bruit	**Vecteurs d'entrée avec bruit gaussien d'écart-type** $\sigma=\sqrt{0.2}$	**Vecteurs d'entrée avec bruit gaussien d'écart-type** $\sigma=\sqrt{0.4}$
Nombre de bonnes classifications sur une base de 30 vecteurs	30	29	27
Répartition des réponses du réseau			

Tableau 3-1 Statistique des réponses du réseau de neurones en phase de validation pour la reconnaissance du chiffre 1 sur une base de 30 vecteurs d'entrée.

La figure 3-9 donne la représentation des chiffres cible et bruités. Un seul de ces chiffres n'a pas été reconnu (image b), les autres chiffres (images c et d) ont été correctement identifiés malgré leur très forte dégradation.



(a) Image cible (b) Image non reconnue par le réseau (c) Image reconnue par le réseau (d)Image reconnue par le réseau

Figure 3-9 Représentation des images cible et bruitées appliquées au réseau en phase de validation

Nous avons constaté (*cf.* §2.3.3.3) que lorsque les poids sont perturbés par la méthode Full Perturbation, des écarts pouvaient apparaître entre les valeurs calculées en fin d'apprentissage pour un même poids. La figure 3-10 illustre bien la situation en donnant la répartition de la valeur finale des dix premiers poids du réseau (étoiles) autour de leur valeur moyenne (cercle). Cela rappelle que la convergence du processus d'apprentissage est basée sur le critère d'erreur moyenne c'est à dire que la fin de l'apprentissage peut être atteint pour des valeurs d'un même poids assez disparates entre plusieurs apprentissages convergents. Le fait d'observer la valeur finale des poids ne permet pas de statuer sur la « qualité » d'un apprentissage. L'exemple de la porte AND de la figure 2-14 du chapitre 2 rappelle que le réseau traite le problème de classification en ajustant ses coefficients pour trouver une droite qui sépare au mieux les vecteurs d'entrée. La répartition des poids sur la figure 3-10 démontre bien qu'il existe une infinité de cœfficients qui permettent de positionner la droite de séparation.

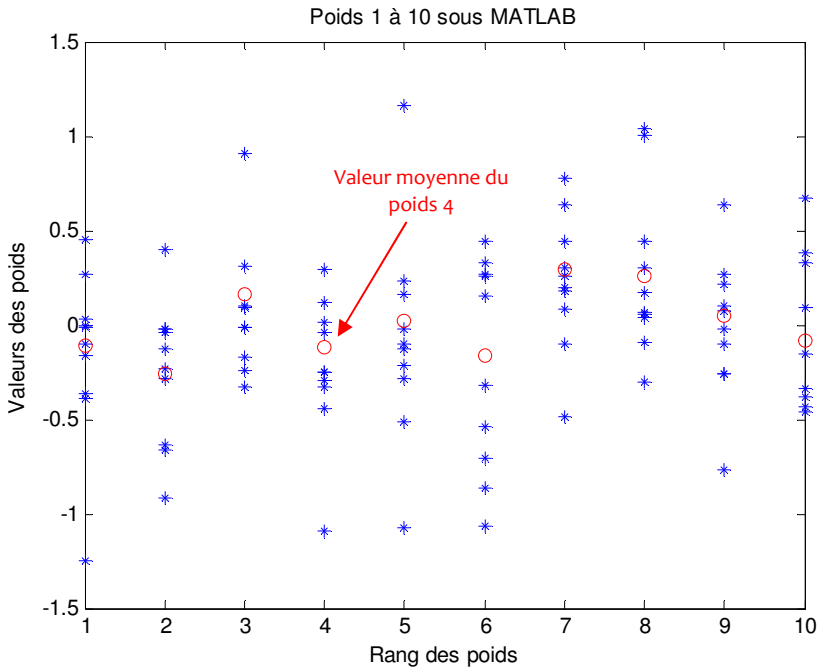


Figure 3-10 Valeurs calculées des 10 premiers poids pour 30 apprentissages convergents avec un seuil de convergence égal à 0.01.

L'exemple de la porte AND du chapitre 2 a également montré que si la moyenne prise pour chaque poids synaptique est calculée sur un nombre assez grand d'apprentissages alors cette

valeur se rapproche des solutions calculées par la technique séquentielle en Weight Perturbation. C'est ce que nous allons vérifier maintenant.
La figure 3-11 donne une superposition des valeurs moyennes pour 30 apprentissages en Full Perturbation avec les valeurs finales calculées par la technique séquentielle. Cette superposition permet alors de confondre les valeurs moyennes pour les deux méthodes de perturbation.

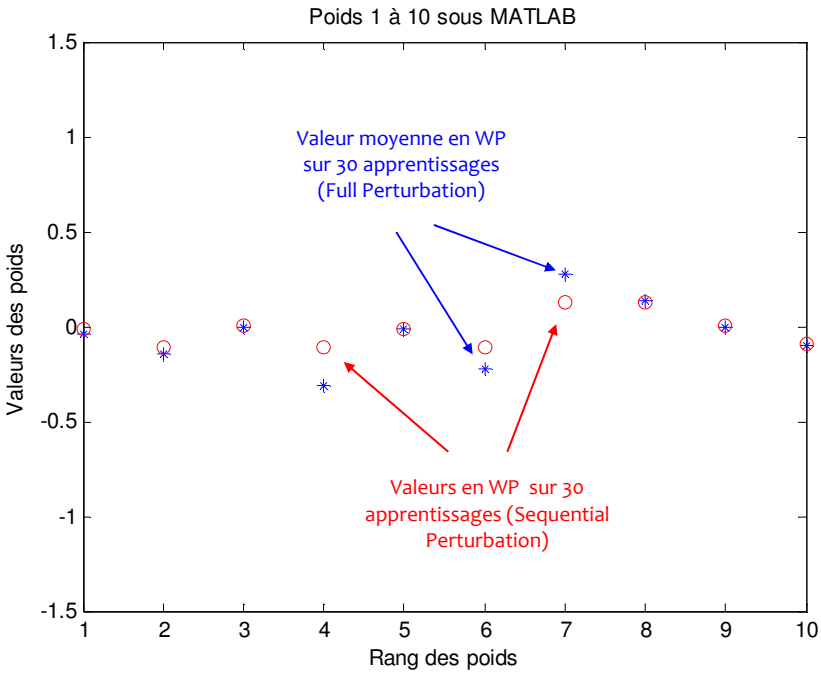


Figure 3-11 Valeurs moyennes calculées des 10 premiers poids (étoiles) en WP Full Perturbation (30 apprentissages) avec un seuil de convergence de 0.01 et valeurs calculées par la technique séquentielle (cercles).

Nous avons décrit le comportement dynamique de l'algorithme d'apprentissage du réseau de neurones sous MATLAB. L'algorithme Weight Perturbation et sa méthode Full Perturbation permet d'observer, comparativement à la méthode séquentielle, une vitesse de convergence plus élevée : la méthode Full Perturbation requiert globalement 10 fois moins de cycles d'apprentissage. Les valeurs finales des poids qui traduisent l'équation de l'hyperplan séparateur peuvent prendre des valeurs différentes d'un apprentissage à un autre et démontre qu'il n'existe pas de solution unique. Ceci montre toute la souplesse d'un réseau de neurones à converger pour des valeurs de poids non prédéterminées a priori. Cet avantage sera pour nous déterminant dans la conception d'un réseau à l'aide de circuits analogiques.

L'étape suivante consiste à décrire le réseau feedforward et son algorithme en vue d'une implémentation sur un circuit analogique. Cette étape nécessite l'utilisation du langage de description matérielle VHDL-AMS.

3.4 Transposition du modèle de l'application de MATLAB vers le langage VHDL-AMS

Nous abordons maintenant la phase de partitionnement de notre application. Il s'agit de décomposer l'architecture du réseau sous sa forme fonctionnelle en proposant une description comportementale. Cette phase doit nous mener progressivement à une description du réseau d'un niveau d'abstraction plus proche du circuit pour finalement aboutir à une sélection d'architectures.

Cette phase de description nécessite néanmoins que des précautions soient prises pour s'assurer que le modèle décrit sous MATLAB soit bien retranscrit par le modèle qui sera développé en VHDL-AMS. L'objectif de ce travail est de confronter les analyses d'une simulation du modèle comportemental du réseau et de son algorithme d'apprentissage sous MATLAB à celles du modèle décrit sous VHDL-AMS. Ceci fera alors de ce modèle comportemental une base de développement solide et fiable pour la poursuite de la phase de dimensionnement.

La plateforme qui décrira et simulera le réseau de neurones en VHDL-AMS sera le logiciel *SMASH Dolphin Integration v5.6*.

3.4.1 Description du modèle en VHDL-AMS

Le modèle du réseau et de son algorithme d'apprentissage décrit sous Matlab correspond à une vue purement comportementale au niveau *signal-flow*. Ce modèle est transposé en VHDL-AMS par une interconnexion de fonctions décrivant le comportement global de l'application (figure 3-12). La description du mécanisme d'apprentissage du réseau sous VHDL-AMS donne une organisation modulaire et orientée conception. Chaque fonction du réseau et de son processus d'apprentissage est codée dans un objet. L'objet qui séquence les opérations est l'objet Top, il est chargé de présenter un à un les vecteurs d'entée au réseau (objet Network) et les vecteurs cibles à l'algorithme d'ajustement des poids (objet WP). Pour permettre de dresser un comparatif fiable, il est primordial que les vecteurs d'entrée et cibles présentés au réseau soient identiques pour les deux descriptions faites sous MATLAB et VHDL-AMS. Pour cela, les matrices générées par MATLAB sont enregistrées dans des fichiers exportables et exploitables par la description codée en langage VHDL-AMS. L'objet WP renvoie les nouvelles valeurs calculées des poids au réseau en fonction du niveau électrique de sa sortie. A noter que l'algorithme d'apprentissage construit sa matrice de poids synaptiques à partir des valeurs initiales générées par MATLAB. Ce point est particulièrement important et nous verrons plus loin que les valeurs de poids au départ influencent fortement l'évolution de ces derniers durant la phase d'apprentissage, il est donc primordial d'en tenir compte si l'on tient à dresser un comparatif réaliste des deux méthodes. Les vecteurs d'entrée et cibles sont stockés dans des fichiers et c'est l'objet Top qui est donc chargé de les prélever. L'objet Top crée également un fichier dans lequel sont enregistrées les valeurs des poids synaptiques après apprentissage du réseau pour un post traitement externe (analyse des résultats en vue du bilan de comparaison dans l'évaluation de la ressemblance des caractéristiques d'apprentissage).

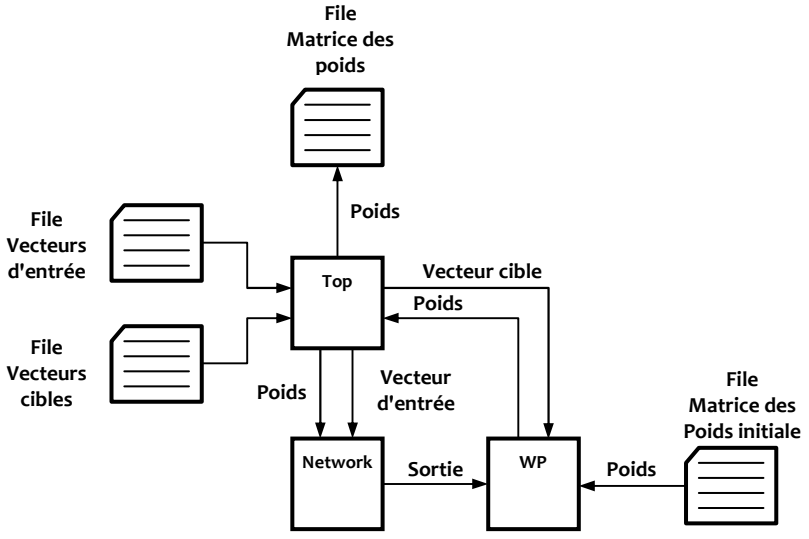


Figure 3-12 Synoptique de la description VHDL-AMS.

Afin de statuer sur l'équivalence des modèles, deux tests sont effectués. Le premier test applique la même séquence de perturbation de poids synaptiques aux deux modèles décrits respectivement sous MATLAB et en VHDL-AMS. Le second test, utilise ensuite une séquence de perturbation propre à chaque modèle.

3.4.2 Tests d'équivalence lorsque la séquence de perturbation est identique

Le travail qui suit vise donc à s'assurer que les modèles décrits à la fois sous MATLAB et SMASH pour le VHDL-AMS se comportent de façon identique selon une démarche bien établie.

L'approche mise en place pour démontrer l'équivalence entre les modèles d'algorithmes décrits sous MATLAB et en VHDL-AMS est menée en posant des critères d'équivalence basés sur la ressemblance des évolutions du TPMSE ou fonction de coût (noté FC) et des poids synaptiques du réseau pour les deux modèles. Cette approche est détaillée figure 3-13. Elle montre qu'à partir de la base des vecteurs d'apprentissage et de l'ensemble des signaux de perturbation, chacune des descriptions (MATLAB et SMASH) fournit, après apprentissage, un lot de solutions aux poids synaptiques (w_1 pour MATLAB et w_2 pour SMASH) ainsi que la progression de la fonction de coût (FC_1 pour MATLAB et FC_2 pour SMASH). Le différentiateur, nécessaire à la mise en place du premier critère de comparaison, est sollicité pour mesurer l'écart entre les deux évolutions de la fonction de coût.

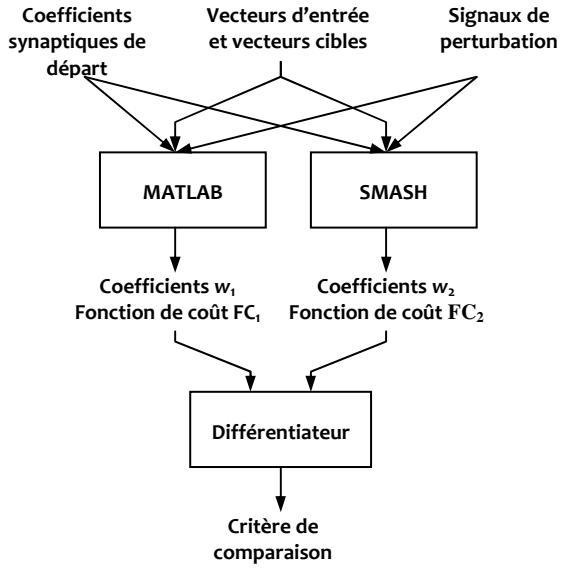


Figure 3-13 Diagramme des opérations du test d'équivalence

Pour ce premier test de comparaison, la base des vecteurs d'apprentissage (entrée et cibles), la matrice des signes aléatoires pour la perturbation des poids synaptique et la matrice des coefficients synaptiques initiaux sont fixées et communes aux deux descriptions. L'évolution au cours de l'apprentissage de l'un des poids synaptiques (poids n°1 : soit alors w_1 pour MATLAB et w_2 pour SMASH) ainsi que la progression du TPMSE (c'est-à-dire de la fonction de coût : FC_1 pour MATLAB et FC_2 pour SMASH) sont présentées figure 3-14.

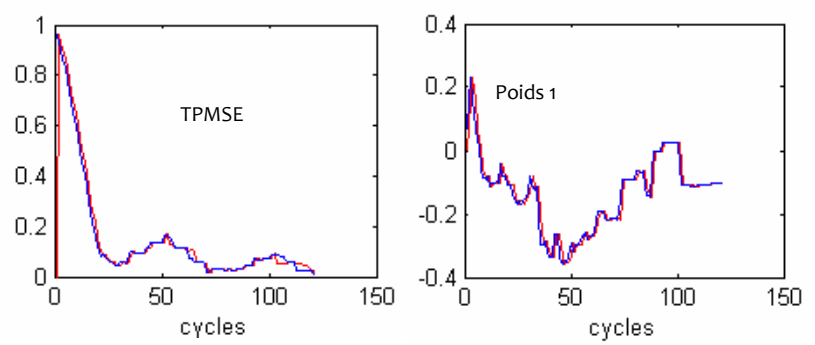


Figure 3-14 Diagramme des opérations du test d'équivalence

Dans les deux cas, la représentation des tracés est très semblable. On note par ailleurs que l'algorithme converge exactement en 122 cycles dans les deux simulations, que l'écart type entre les courbes du TPMSE est de 0,087 et que l'écart type entre les courbes du poids n°1 est de 0,025. Ces chiffres confirment donc l'équivalence des deux modèles selon le précepte que les mêmes causes donnent les mêmes effets. Néanmoins, l'algorithme WP nécessite un processus aléatoire pour la perturbation des poids. Il est impératif de vérifier le comportement de ces algorithmes lorsque les générateurs aléatoires des deux simulateurs sont indépendants.

3.4.3 Tests d'équivalence lorsque la séquence de perturbation est indépendante

Une première variante de l'algorithme WP consiste à modifier l'ensemble des poids synaptiques simultanément (*Full WP method*). Un protocole spécifique de comparaison a été fixé et les résultats sont présentés ci-dessous. Pour ce second test, la base des vecteurs d'apprentissage (entrées et cibles) et la matrice des coefficients synaptiques initiaux sont fixées et communes aux deux descriptions. Mais la matrice des signes des perturbations est propre à chaque simulation. La figure 3-15 donne l'allure du TPMSE et du poids n°1 sous MATLAB et SMASH. Pour les deux grandeurs, les évolutions suivent des trajectoires bien distinctes. L'algorithme converge en 139 cycles sous MATLAB et en 128 cycles sous SMASH. Les écarts relevés montrent l'influence du tirage des signes de la perturbation sur l'évolution des courbes. Il est d'ailleurs important de rappeler qu'avec la méthode Full Perturbation les poids convergent en des points différents d'une simulation à une autre. La simple comparaison des évolutions des deux grandeurs ne suffit plus à statuer sur l'équivalence des modèles.

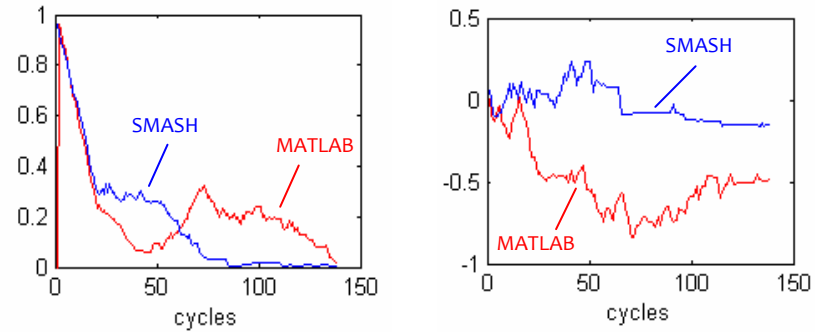


Figure 3-15 Evolution du TPMSE (à gauche) et du poids 1 (à droite) sous MATLAB et en VHDL-AMS lorsque la matrice de perturbation est indépendante

Pour cela, une autre expérience s'intéresse à la valeur moyenne des poids synaptiques après plusieurs apprentissages. Une série de 30 simulations, toutes convergentes et menées par des sources aléatoires de perturbation indépendantes, a été réalisée sous MATLAB puis sous SMASH. La valeur moyenne de chaque poids synaptique sur l'ensemble de la série a été calculée. Chacune de ces valeurs moyennes est représentée sur la figure 3-16 par un niveau de couleur allant du noir

pour les valeurs négatives au blanc pour les valeurs positives. Cette approche montre l'influence et la valeur prise par les poids selon les zones de l'image à « filtrer ». Les zones obéissent de la même façon sous MATLAB et SMASH avec un groupe de poids (valeur moyenne) globalement négatif (autour de -0,1) pour les pixels du fond noir et positifs (autour de +0,2) pour les pixels du chiffre (barre verticale blanche du chiffre 1). Il en ressort que l'intensité du potentiel d'activation est amplifiée par les valeurs « fortes » des poids synaptiques présents dans les zones du chiffre jusqu'à rendre ce potentiel supérieur au seuil de la fonction d'activation. Dans les deux cas (MATLAB et SMASH), le neurone réagit alors à une détection du chiffre 1. Dans les zones sombres, le potentiel d'activation est affaibli par les valeurs des poids, ce qui rend le neurone inapte à réagir aux excitations des pixels concernés. Le fait d'observer un étalement des valeurs des poids calculés après apprentissage nous a poussé, dans cette expérience, à travailler avec la moyenne de ces valeurs (traduite ici en niveaux de gris) pour établir un critère de comparaison.

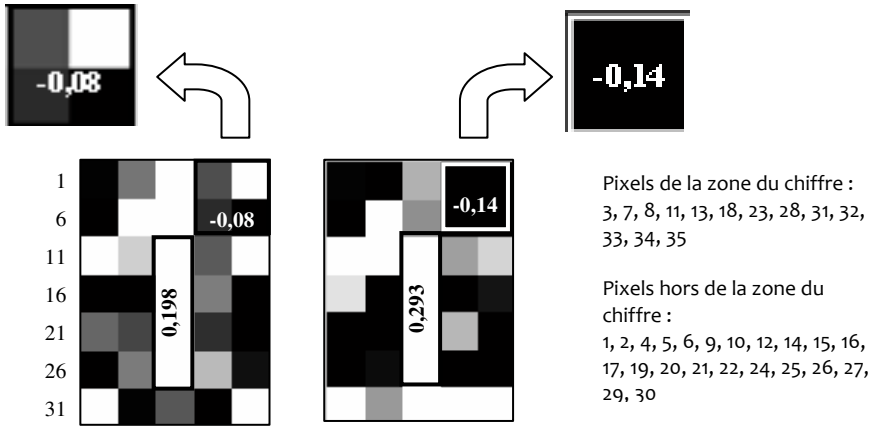


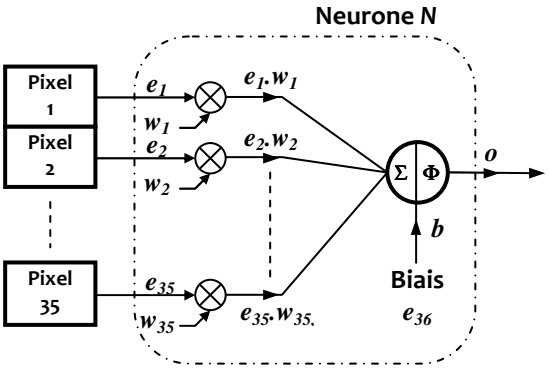
Figure 3-16 Intensité des poids synaptiques sous Matlab (à gauche) et Smash (à droite)

La comparaison de ces deux modèles (Matlab et Smash), dès lors que les sources aléatoires des signaux de perturbation sont propres à chacune des descriptions, conduit à des écarts notamment entre les valeurs calculées des poids synaptiques en fin d'apprentissage. Néanmoins, cette situation a tendance à fortement diminuer lorsque l'on travaille sur un nombre suffisant d'apprentissages. Il en ressort alors un comportement algorithmique des deux modèles tout à fait comparable avec globalement des valeurs de poids qui tendent vers des minima ou maxima. Il est évident que deux processus stochastiques indépendants, ici sollicités lors de la technique de perturbation parallèle pour « proposer » un chemin dans l'évolution des poids synaptiques tout au long de l'apprentissage, ne peuvent conduire qu'à des résultats statistiquement ressemblants.

Cette partie nous a permis de vérifier l'équivalence des modèles comportementaux à haut niveau d'abstraction du réseau de neurones et de son algorithme d'apprentissage. Cette vérification nous donne, avec un degré de confiance, l'assurance que le modèle comportemental VHDL-AMS répond aux critères de conception fonctionnels. Ce modèle peut maintenant servir de prototype pour décrire les fonctions internes du réseau.

3.5 Conception électronique d'une cellule neuronale

Nous souhaitons proposer une architecture fonctionnelle puis structurelle de notre réseau de neurones. La figure 3-17 rappelle le schéma de connexion entre les pixels de la matrice et les entrées du réseau.



Sortie $o_k = \phi(p) = \phi(\frac{1+th(p)}{2})$

on pose généralement : $p = e_{36}.b_k + \sum_{i=1}^{35} e_i.w_i$

qui devient avec : $e_{36} = 1 : p = b_k + \sum_{i=1}^{35} e_i.w_i$

Figure 3-17 Schéma de connexion du réseau à la matrice de pixels.

Les grandeurs e_i (i=1, …, 35) étant partagées entre tous les neurones, ces grandeurs seront alors des tensions. Les grandeurs ($e_i w_i$) étant sommées pour donner le potentiel d'action p du neurone, ces grandeurs seront des courants. Le potentiel d'activation sera par conséquent élaboré simplement au moyen d'une interconnexion. Les circuits à mode courant et en particulier les circuits à courants commutés ont gagné en intérêt dans le domaine de l'électronique analogique notamment pour leur utilisation à basse tension et en hautes fréquences sans qu'il y ait limitation dans la dynamique de sortie [Tou-90a]. Dans le cadre de notre travail, l'apport de ces circuits sera mis à profit pour compléter le dispositif à réseau de neurones.

3.5.1 Partitionnement du réseau de neurones

L'architecture fonctionnelle du neurone est présentée sur la figure 3-18. Cette architecture est partielle et se limite à la représentation des branches synaptiques 1, 2 et 36 (la branche synaptique du biais).

Le réseau reçoit sur son entrée le signal cible (t) et la partie neurone reçoit en entrée les signaux du capteur (e_1 à e_{36}) qui représentent les niveaux de tension de chaque pixel. Ces signaux sont convertis en courants à partir des blocs à transconductance (G_t et G_1 à G_{36}). Les courants d'entrées (I_1 à I_{36}) sont ensuite multipliés par les courants synaptiques (image des poids du réseau) dans les blocs à synapses S_1 à S_{36} puis convergent en un nœud vers le bloc de la fonction d'activation (ϕ) pour constituer le potentiel d'action du neurone (courant I_P). Le potentiel d'action est transformé par la fonction d'activation (ϕ) en un signal de sortie (o). La tension de sortie (o) est distribuée à l'ensemble branches synaptiques. Cette tension est alors localement à chaque branche synaptique transformée en un courant I_o. Le courant I_o alimente l'ensemble des blocs W_1 à W_{36} qui assurent la génération des courants synaptiques I_{w1} à I_{w36}. Ces blocs implémentent l'ensemble des fonctionnalités de l'algorithme Weight Perturbation. Il est donc nécessaire d'y appliquer le courant de cible I_t qui sera comparé au courant I_o. C'est à partir de cette comparaison que les courants synaptiques sont ensuite calculés. Ces courants doivent être maintenus constants le temps d'un cycle d'apprentissage. Ceci pose le problème du stockage temporel du courant et par conséquent l'utilisation de fonctions à mémoire de courant. Une première réponse a été donnée par la cellule à courant commutée de type S2I qui recopie le courant de son entrée vers sa sortie par commutation [Vau-00]. Cette propriété est exploitée pour maintenir un courant constant durant un cycle entier en phase d'apprentissage du réseau de neurones. Il est donc indispensable de mesurer les limites et performances d'un tel circuit. L'objectif de ce travail est de tester le modèle VHDL-AMS des blocs algorithmiques W_1 à W_{36} et en

particulier des cellules à mémoire de courant. Le modèle pourra ensuite être inséré dans l'architecture complète.

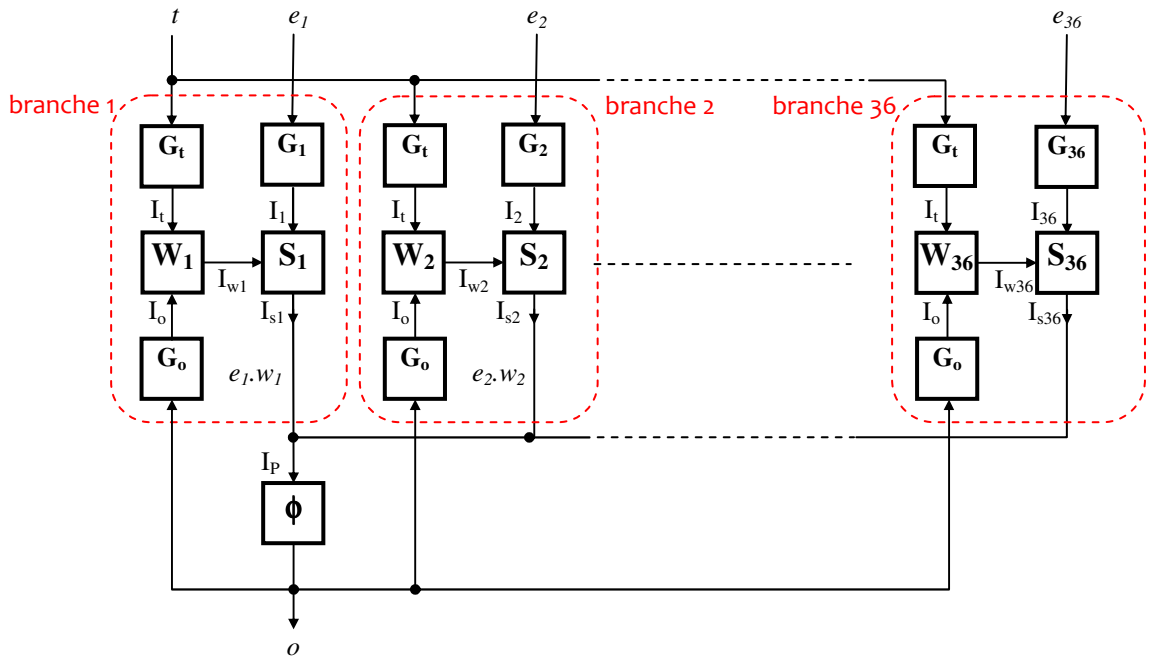


Figure 3-18 Architecture fonctionnelle du neurone

Description des blocs :

- Bloc à transconductance : G_i

Le bloc à transconductance est constitué d'un amplificateur à transconductance (OTA). Chacune des tensions du réseau (tension cible, tension de la matrice des pixels et tension de sortie du neurone) sont convertis en courant par ce bloc.

- Bloc à synapses : S_i

Le bloc à synapses réalise le produit des courants de ses deux entrées (courant d'entrée I_i et courant du poids synaptique I_{wi}). Ce bloc sera constitué d'un multiplieur analogique travaillant en mode courant [Mic-03, Sun-08].

- Bloc algorithmique : W_i

Le bloc contient le processus d'apprentissage de l'algorithme WP pour chaque poids synaptique. L'information synaptique étant un courant, ce bloc contient également un dispositif de mémorisation des courants de poids.

- Bloc à fonction d'activation : ϕ

Le bloc à fonction d'activation doit assurer la fonction de seuil. Une topologie à paire différentielle permet de bénéficier d'une fonction de transfert de type sigmoïdale [Dio-00].

L'utilisation de circuits analogiques et le côté non idéal des composants peuvent introduire des défauts sur les caractéristiques d'entrée/sortie de chaque bloc. Par exemple, des écarts entre le courant synaptique calculé par l'algorithme et le courant réellement acheminé au bloc à synapse peuvent être à l'origine de dysfonctionnement (ralentissement ou même non convergence) dans le processus d'apprentissage. Nous cherchons donc à mesurer l'impact qu'auraient de telles variations sur le comportement algorithmique du réseau. Nous espérons néanmoins que la rétroaction du processus de correction des poids synaptiques permettra de compenser les défauts intrinsèques de chaque composant. Le travail qui suit vise donc à vérifier cela.

Nous allons donc commencer par proposer une description structurelle du bloc algorithmique W_i et vérifier ensuite comment évolue le processus d'apprentissage.

3.5.1.1 Sélection de l'architecture du bloc algorithmique W

Un réseau de neurones comporte un nombre déterminé de poids synaptiques dont les valeurs évoluent au rythme des cycles d'apprentissage. Les vecteurs d'informations des poids synaptiques étant des courants, il apparaît alors la contrainte du maintien de ces courants durant chaque cycle d'apprentissage. Cette opération de mémorisation peut être accomplie par un dispositif constitué de cellules à courants commutés CR-S2I [Mic-03]. Le dispositif de mémorisation s'insère dans chaque branche d'entrée d'un neurone après le bloc d'apprentissage (figure 3-19). Il assure la fonction de mémorisation des courants synaptiques (poids synaptique) calculés par l'algorithme WP. L'algorithme fournit en effet deux courants : le courant de perturbation dont la valeur est constante mais dont le signe est soit positif soit négatif et le courant de correction à apporter au poids dont la valeur et le signe sont variables. Ces valeurs sont modifiées à chaque cycle d'apprentissage de l'algorithme (*cf.* §2.3.2.3). Le dispositif doit donc fournir à chaque cycle *k* et à chaque synapse du neurone un courant (poids synaptique) stable et fidèle aux équations de correction des poids :

$I_{wi}(k) = I_{wi}(k-1) + I_{PERT\,i}(k)$, si l'algorithme est en phase de perturbation
$I_{wi}(k) = I_{wi}(k-1) + I_{\Delta wi}(k)$, si l'algorithme est en phase de correction

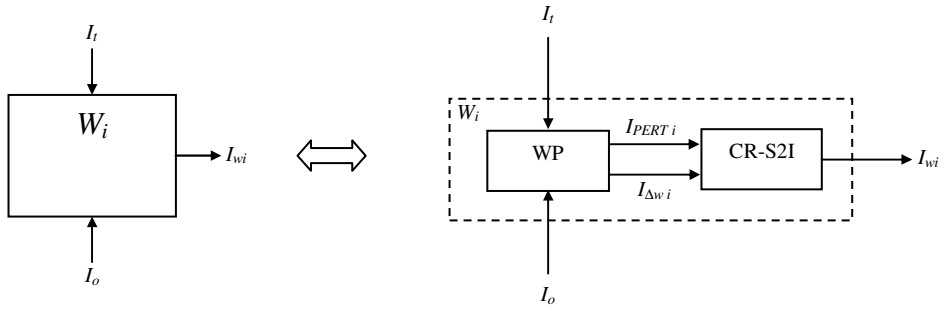


Figure 3-19 Le bloc synaptique avec son algorithme d'apprentissage et son dispositif de mémorisation des poids synaptiques

3.5.1.2 Présentation du dispositif de mémorisation des courants synaptiques CR-S2I

Le dispositif de mémorisation doit maintenir les courants synaptiques calculés par le bloc W_i après chaque phase de perturbation ou de correction. La figure 3-20 illustre le synoptique du dispositif de mémorisation (les signaux de commande des commutateurs ne sont pas représentés). Quatre cellules de mémorisation à courant commutés (S2I) sont nécessaires. Une cellule mémorisera le courant de perturbation, nous la nommerons la cellule $S2I_{PERT}$. Deux autres cellules mémoriseront le courant synaptique, nous les appellerons $S2I_{W1}$ et $S2I_{W2}$. Enfin la quatrième cellule est chargée d'échantillonner et de bloquer le courant à une valeur constante en sortie du dispositif CR-S2I. Nous nommerons cette cellule $S2I_{S\&H}$. Un descriptif détaillé du fonctionnement d'une cellule S2I élémentaire est donné en pages annexes.

La figure 3-20(a) donne la vue décomposée du dispositif CR-S2I. Les cellules S2I fonctionnent ici en régime continu c'est-à-dire qu'aucune commutation n'intervient à l'intérieur de ces blocs. Le courant entrant dans une cellule S2I est donc instantanément recopié sur sa sortie. Le fonctionnement peut être décomposé selon trois phases principalement et la position des cinq commutateurs Sw_1, Sw_2, Sw_3, Sw_4, et Sw_S y jouent un rôle primordial.

① Phase de recopie du courant de perturbation : Sw_1, Sw_S fermés

Durant cette phase, le courant de perturbation I_{PERT} est mémorisé dans la cellule $S2I_{PERT}$ pour être ensuite ajouté au courant synaptique issu de la cellule $S2I_{W2}$ et enfin débité dans le bloc $S2I_{S\&H}$. La fonction d'échantillonnage et blocage permet de maintenir le courant en dehors de la phase 1. Le courant de sortie du dispositif est alors égal au courant de perturbation à l'itération *k* à : $I_W(k) = I_W(k-1) + I_{PERT}(k)$

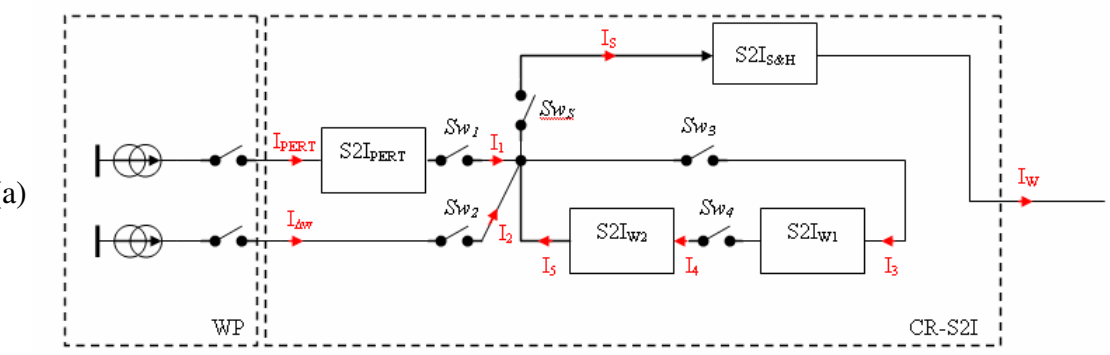
② Phase d'ajustement du courant synaptique : Sw_2, Sw_3 fermés

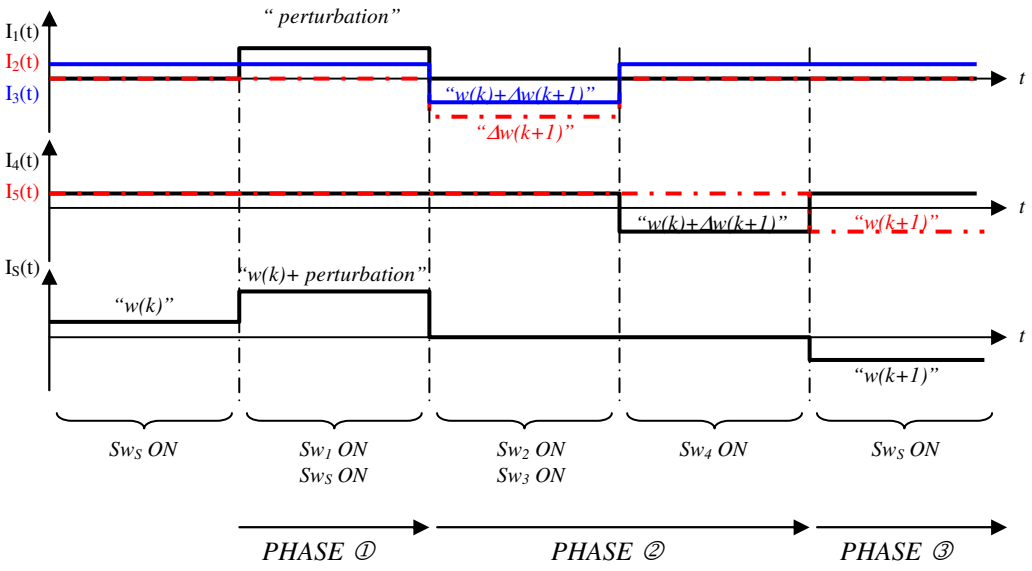
Durant cette phase, un courant $I_{\Delta W}$ est transmis à la cellule $S2I_{W1}$ en mode écriture auquel vient s'ajouter le courant synaptique du cycle d'apprentissage en cours mémorisé dans la cellule $S2I_{W2}$. La somme de ces deux courants forme le nouveau courant synaptique du cycle d'apprentissage à venir. Ce courant synaptique est mémorisé dans la cellule $S2I_{W1}$ pour être ensuite recopié dans la seconde cellule $S2I_{W2}$ et seul Sw_4 est alors fermé.

③ Phase de recopie du courant synaptique : Sw_S fermé

Le courant synaptique mémorisé dans la cellule $S2I_{W2}$ est transmis à la charge par l'intermédiaire de la fonction d'échantillonnage et blocage. Le courant de sortie de la cellule $S2I_{W2}$est alors égal à l'itération *k* à : $I_W(k) = I_W(k-1) + I_{\square W}(k)$

(a)





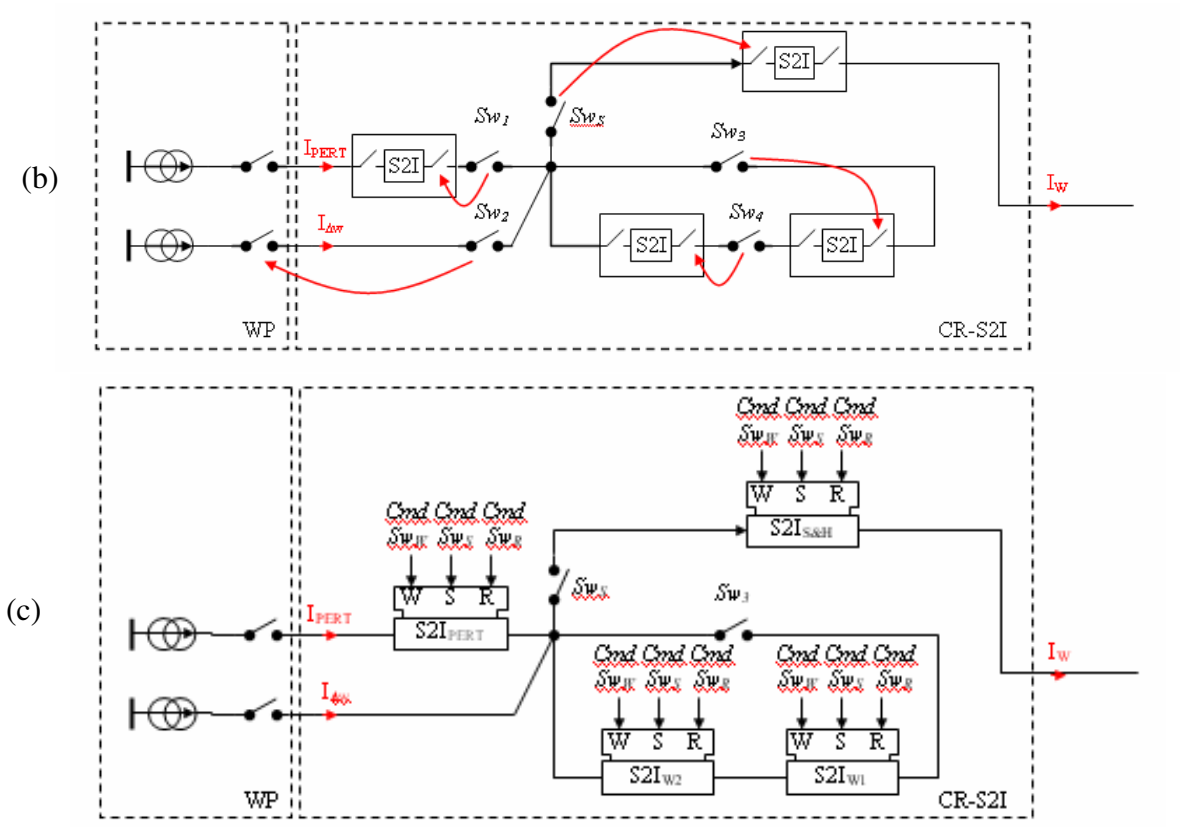


Figure 3-20 Synoptique du dispositif de mémorisation à courant commutés. Les courants I_{PERT} et $I_{\Delta W}$ sont débités par le bloc WP.

La figure 3-20(b) fait apparaître les commutateurs d'écriture et de lecture internes à chaque cellule S2I. La figure met en évidence la redondance de certains commutateurs et la figure 3-20(c) représente le synoptique définitif du dispositif CR-S2I. Seuls les commutateurs Sw_S et Sw_3 assurent la commutation des courants en-dehors des cellules S2I. Les fonctions de commutation restantes sont-elles internes à chacune des cellules où la symbolisation de la commande des commutateurs d'écriture (W), d'échantillonnage (S) et de lecture (R) est ajoutée (voir pages annexes).

3.5.1.3 Présentation des cellules de mémorisation des courants synaptiques S2I

Le choix structurel de la cellule repose sur trois critères principalement ; la précision de recopie, la consommation et l'encombrement de la cellule. Le premier critère implique que le dispositif de mémorisation soit capable de régénérer avec la plus grande précision possible le courant d'entrée sur sa sortie. Le cas idéal se traduit par l'égalité du courant qui rentre dans la cellule au courant qui en sort tel que : $I_I = I_O$. Or, un certain nombre de défauts sont à prendre en considération [Dau-90] donnant lieu à des architectures disposant de modules de correction plus ou moins élaborés [Tou-93, Tan-97]. L'un des problèmes les plus répandus est celui de l'erreur de recopie introduite par l'effet de modulation de longueur de canal du transistor de mémorisation et l'ajout d'un amplificateur opérationnel permet d'atténuer le défaut [Dau-88]. Le dispositif à cascode régulée est également très utilisé pour compenser ce type d'erreur [Tou-90a]. Une autre source d'erreur est celle provoquée par le phénomène d'injection de charges dans les jonctions des transistors (clock feedthrough error) et des algorithmes permettent dans ce cas, par une séquence de commande réalisée sur trois cellules, de propager l'erreur jusqu'à l'annuler [Tou-90b, Tou-94]. Les techniques employées pour améliorer la précision de recopie du courant d'entrée sont donc nombreuses et efficaces mais souvent au prix d'une électronique de

correction qui peut devenir lourde surtout lorsque le dispositif doit être dupliqué et adapté au nombre de cellules neuronales. Enfin, dans le projet de conception qui nous intéresse, un réseau de neurones s'appuie sur les signaux d'entrée et ceux générés par les cellules internes pour élaborer son signal de sortie. Or ces cellules internes au comportement non idéal introduisent une part de distorsion qui est une composante des signaux sur lesquels le réseau base son apprentissage. Il peut donc être envisagé que ces défauts soient pris en compte en phase d'application et que le réseau construit sa réponse en fonction de ces défauts intrinsèques. Ce dernier point sera par conséquent vérifié par simulation. Dans ce cas, il parait judicieux d'orienter la recherche vers une structure légère qui réponde aux exigences de critères de consommation et d'encombrement. En réponse, la cellule S2I remplit la fonction de mémoire de courant avec un nombre limité de composants [Vau-00]. Le modèle simplifié de la cellule S2I est donné sur la figure 3-21. La cellule est construite autour d'un transistor de mémorisation M_{MEM} et trois commutateurs (Sw_{WRITE}, Sw_{SAMPLE} et Sw_{READ}) sont nécessaires pour réaliser les opérations d'écriture, d'échantillonnage et de lecture.

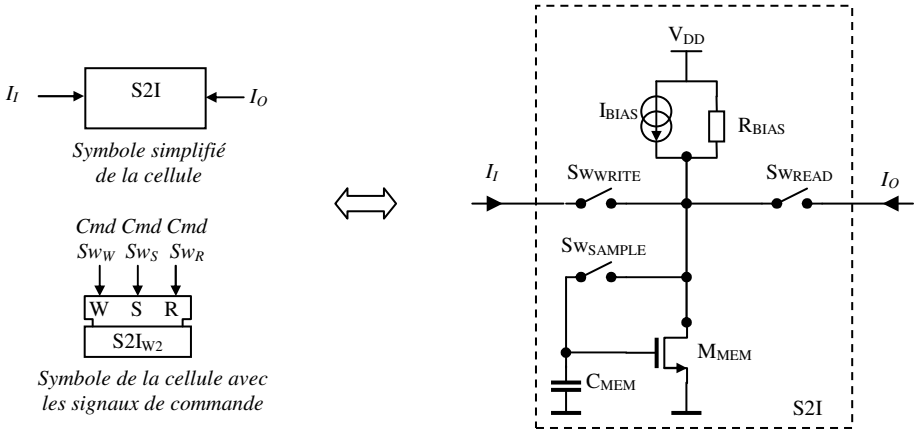


Figure 3-21 Schéma idéalisé de la cellule à courant commuté

La cellule fonctionne donc selon un cycle trois phases :

La phase d'écriture : Sw_{WRITE} = fermé, Sw_{SAMPLE} = fermé, Sw_{READ} = ouvert.
Le courant d'entrée traverse le commutateur Sw_{WRITE} et le transistor M_{MEM} lui-même polarisé par sa tension grille-source fonctionne en régime de saturation. Son courant de drain est donc égal au courant d'entrée additionné au courant de polarisation I_{BIAS}.

La phase d'échantillonnage : Sw_{WRITE} = ouvert, Sw_{SAMPLE} = fermé, Sw_{READ} = ouvert.
Le transistor M_{MEM} est toujours polarisé par sa tension grille-source et maintient par conséquent un courant de drain égal au courant d'entrée additionné au courant de polarisation I_{BIAS}.

.

La phase de lecture : Sw_{WRITE} = ouvert, Sw_{SAMPLE} = ouvert, Sw_{READ} = fermé.
La capacité de mémorisation C_{MEM} garantit les mêmes conditions de polarisation et le courant de drain de M_{MEM} correspond donc au courant de sortie ajouté au courant de polarisation I_{BIAS}. La fermeture du commutateur provoque donc le passage d'un courant égal au seul courant d'entrée.

Remarque : Une description détaillée d'une cellule S2I est donnée en pages annexes de ce chapitre.

3.5.2 Simulation du réseau en phase d'apprentissage

Le réseau est maintenant simulé en procédant à une approche Top-down. Les modèles de haut niveau des blocs décrits dans la partie précédente sont utilisés dans une première étape. On introduira ensuite progressivement des modèles de plus bas niveau pour le bloc CR-S2I. Ceci nous permettra alors d'évaluer l'interaction entre les paramètres de l'apprentissage (évolution des poids synaptiques) et la profondeur des descriptions de chaque modèle de composant du bloc.

Les modèles de blocs sont décrits de la façon suivante :

- Bloc à transconductance : $\boldsymbol{G_i}$

Le modèle du bloc à transconductance est caractérisé par une équation qui traduit le comportement d'une source de courant commandée en tension :

- Bloc à synapses : $\boldsymbol{S_i}$

Le modèle du bloc à synapses décrit, sous la forme d'une équation, le produit terme à terme entre les courants d'entrée et les courants des poids synaptiques. Ce produit est proportionnel à un gain *K* :

$$I_{Si} = K \times I_i \times I_{Wi}$$

- Bloc algorithmique : $\boldsymbol{W_i}$

Le modèle du bloc algorithmique contient l'ensemble du processus d'apprentissage de l'algorithme WP pour chaque poids synaptique. Ce processus décrit fidèlement les étapes détaillées dans le chapitre 2 (*cf.* §2.3.3.3). Le bloc contient également le modèle du dispositif de mémorisation des courants de poids.

Au cycle d'apprentissage *k* :

$I_{wi}(k) = I_{wi}(k\text{-}1) + I_{PERT\ i}(k)$, si l'algorithme est en phase de perturbation
$I_{wi}(k) = I_{wi}(k\text{-}1) + \Delta I_{wi}(k)$, si l'algorithme est en phase de correction

- Bloc à fonction d'activation : $\boldsymbol{\phi}$

Le modèle du bloc à fonction d'activation reprend la forme analytique de l'équation du potentiel d'activation sigmoïdale à tangente hyperbolique appliquée à la somme des courants synaptiques I_P :

$$o = \frac{1 + \tanh(I_P)}{2}$$

Le réseau est simulé en deux étapes. Une première simulation permettra d'observer le comportement global du réseau en phase d'apprentissage sans son dispositif de mémorisation CR-S2I. Il s'agira de vérifier que le modèle comportemental du réseau permet de valider en première approche le partitionnement adopté avec les modèles hauts niveaux précédemment décrits. Dans une seconde simulation, le dispositif de mémorisation CR-S2I sera pris en compte et permettra d'évaluer une architecture réaliste de l'application.

3.5.2.1 Simulation du réseau sans dispositif de mémorisation à cellules CR-S2I

Une première simulation du réseau en phase d'apprentissage du chiffre 1 est réalisée sans insertion de dispositif de mémorisation. Le bloc W_i contient par conséquent uniquement le modèle de l'algorithme WP qui génère les courants synaptiques de perturbation et les courants synaptiques corrigés.

Le réseau doit générer une tension *o* sur sa sortie proche de 900mV lorsque le chiffre 1 est présenté et proche de 100mV dans le cas contraire. Nous interprétons de cette façon une transcription du principe imposant les valeurs nominales des niveaux de décision à 10% et 90% de la valeur pleine échelle ici de 1V (*cf*. §3.1). La présentation au réseau du chiffre 1 est faite une fois sur deux selon le mode du *batch learning*.

La figure 3-22 décrit l'évolution du processus d'apprentissage. Seuls les trois premiers poids synaptiques du réseau sont ici représentés, les tensions de sortie *o* et de cible *t*, et l'allure du PMSE et du TPMSE. Le TPMSE passe sous le seuil de convergence après 66 cycles et les poids sont alors stabilisés sur des plateaux. Le réseau est alors entraîné à reconnaître le chiffre 1 avec un niveau de sortie *o*, en fin d'apprentissage, « calé » sur la valeur cible *t*.

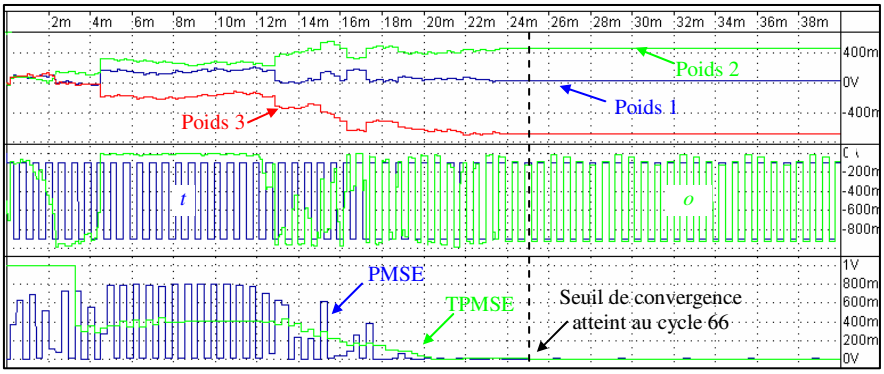


Figure 3-22 Simulation du modèle comportemental du réseau sans dispositif CR-S2I

Cette première étape démontre que l'architecture fonctionnelle choisie et son algorithme nous permet d'atteindre les objectifs d'apprentissage. Ceci nous permet maintenant d'aborder l'analyse *Top-Down* en descendant dans la hiérarchie des niveaux d'abstraction jusqu'à introduire le modèle structurel du dispositif de mémorisation CR-S2I dans le bloc W_i.

3.5.2.2 Simulation du réseau avec dispositif de mémorisation à cellules CR-S2I

Le dispositif de mémorisation est maintenant inséré dans chaque branche synaptique du réseau de neurone. La description du modèle du dispositif sera progressivement affinée tout en maintenant la description des autres blocs (bloc à transconductance, à synapses et à fonction d'activation) aux niveaux comportementaux utilisés jusqu'à présent. Cette analyse a deux objectifs. Il s'agit d'une part de vérifier la fonctionnalité et de mesurer les limites et d'autre part, d'en déduire les contraintes et les spécifications du dispositif.

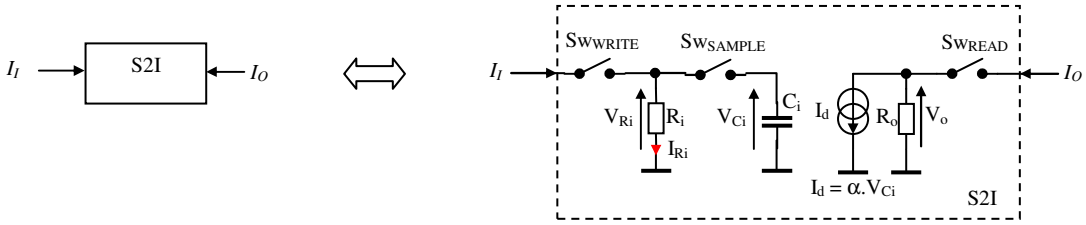
On dispose de trois niveaux de description comportementale de la cellule S2I. Nous les nommerons distinctement : modèle comportemental de niveaux 1, 2 et 3.

Ainsi qu'un niveau de description structurelle que nous nommerons à son tour : modèle structurel de niveau 1.

Le détail de ces descriptions est présenté dans ce qui suit.

Le modèle comportemental de niveau 1 :

Le modèle comportemental de niveau 1 assure la fonction de recopie du courant d'entrée sur la sortie en implémentant un « process » codé en VHDL-AMS. Le code du modèle a été ajouté dans les pages annexes. Une représentation équivalente à l'aide de composants discrets est donnée ci-dessous.



Le modèle comportemental de niveau 1 de la cellule comprend :

- une résistance d'entrée R_i ;
- une capacité de mémorisation C_i ;
- une source de courant idéale image du courant d'entrée I_I ;
- une résistance de sortie R_o ;
- trois commutateurs idéaux Sw_{SAMPLE}, Sw_{WRITE} et Sw_{READ} ;

A la fermeture des commutateurs Sw_{SAMPLE} et Sw_{WRITE}, l'image du courant d'entrée de la cellule détermine, à un coefficient de transconductance α près, la valeur du courant de la source interne I_d telle que :

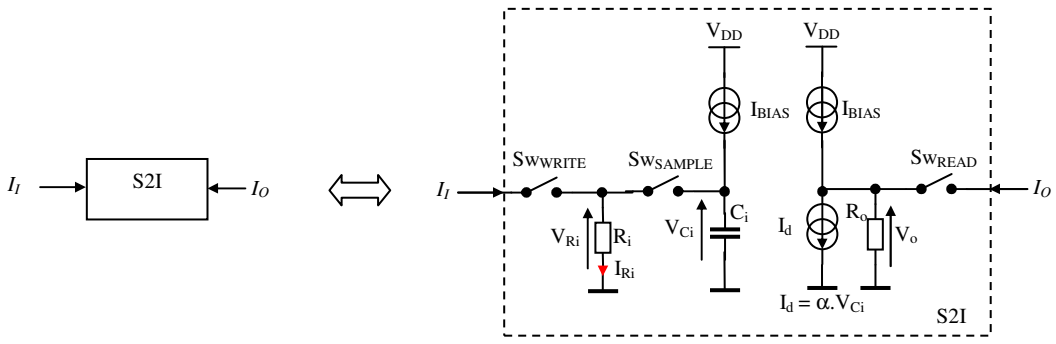
$$I_d = \alpha.V_{Ci}$$

Notons que la tension V_{Ri} est une tension résultante qu'il faudra gérer mais qui n'agit pas directement sur le principe de fonctionnement. La cellule possède une résistance de sortie R_o. Lorsque le commutateur Sw_{READ} est fermé, la valeur du courant en sortie de la cellule sera égale à :

$$I_o = I_d + V_o/R_o$$

La tension V_o est de même nature de V_{Ri} mais qui dépend aussi de la charge connectée à la cellule S2I.

Le modèle comportemental de niveau 2 :

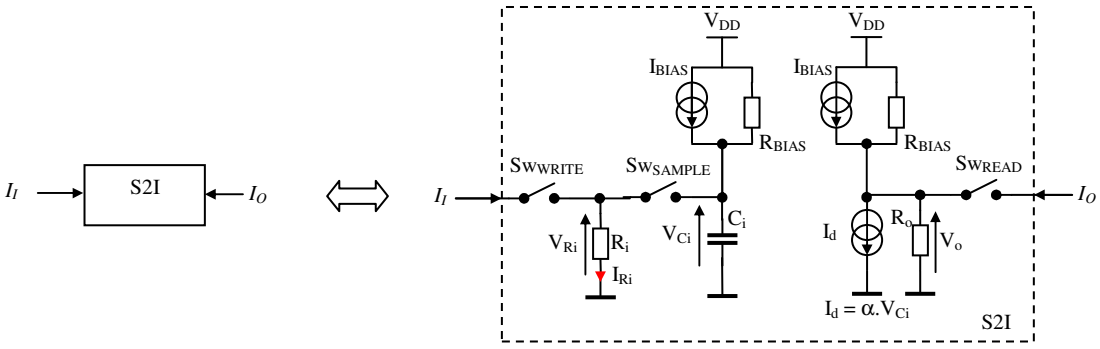


Le modèle comportemental de niveau 2 de la cellule est complété d'une source de courant idéale qui simule le courant de polarisation I_{BIAS} de la cellule à l'entrée de la cellule en mode d'échantillonnage (commutateur Sw_{SAMPLE} fermé).
Ce modèle permet de prendre en compte les limites de la cellule lorsque le courant d'entrée est négatif et inférieur au courant de polarisation. On observe en effet que :

Lorsque : $I_I < 0$ et $|I_I| > I_{BIAS}$ alors $I_{Omax} = I_{BIAS}$

Cette limitation est illustrée en pages annexes par la simulation de la plage de fonctionnement de la cellule S2I (*cf.* §6.2).

Le modèle comportemental de niveau 3 :

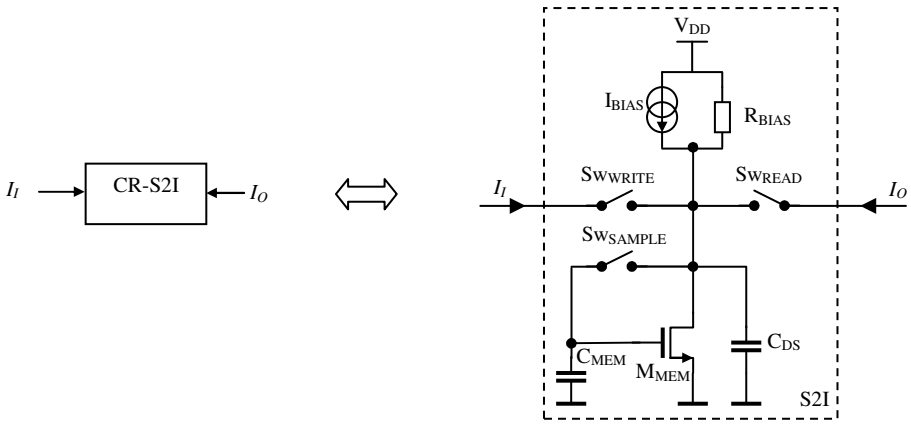


Le modèle comportemental de niveau 3 de la cellule introduit une résistance interne R_{BIAS} à la source de courant de polarisation. Ce modèle apporte donc un niveau de précision supplémentaire en prenant en compte l'impact des tensions de fonctionnement de la cellule (tension d'alimentation V_{DD} et de sortie V_o) sur la valeur de la source de courant I_{BIAS}.

Lorsque le commutateur Sw_{READ} est fermé, la valeur du courant en sortie de la cellule sera égale à :

$$I_o = I_I + V_o/R_o + (V_{DD} - V_o)/R_{BIAS}$$

Le modèle structurel de niveau 1 :



Le modèle structurel de niveau 1 de la cellule comprend :

- un transistor MOS-FET M_{MEM} avec sa capacité drain-source C_{DS} ;

- une source de courant idéale qui simule le courant de polarisation I_{BIAS} de la cellule pour l'ensemble des modes de fonctionnement et sa résistance interne R_{BIAS};
- un condensateur de mémorisation C_{MEM} ;
- trois commutateurs idéaux Sw_{SAMPLE}, Sw_{WRITE} et Sw_{READ} ;

Simulation avec le modèle comportemental de niveau 1 de la cellule S2I

Une première simulation utilise le modèle comportemental de niveau 1 de la cellule S2I dans le dispositif CR-S2I. La figure 3-23 donne l'évolution des paramètres du réseau. L'algorithme d'apprentissage converge en 104 cycles.

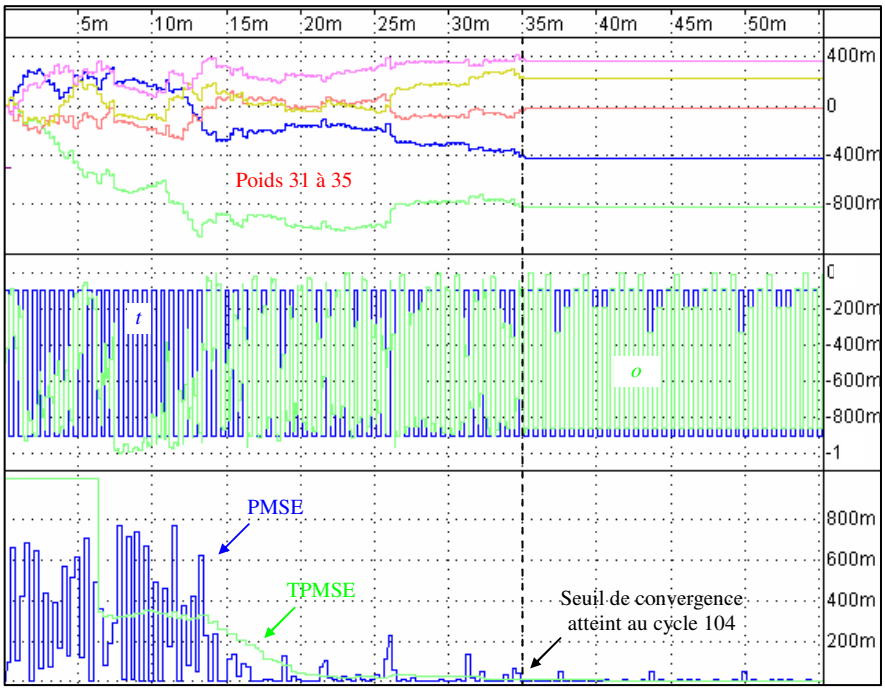
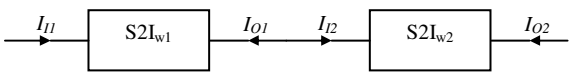


Figure 3-23 Simulation du modèle comportemental du réseau avec dispositif CR-S2I à cellules S2I comportemental de niveau 1

L'observation de l'allure de l'ensemble des poids (ici seuls les poids 31 à 35 ont été représentés) durant le processus d'apprentissage permet de relever les valeurs maximales atteintes. Ces valeurs sont calculées par l'algorithme (bloc WP) et prennent des amplitudes particulièrement importantes (on relève sur la figure 3-23 une amplitude maximale proche de -1A !). Ces valeurs sont critiques et détermineront les valeurs du courant I_{BIAS} de la source interne aux cellules S2I. Une simulation, donnée en pages annexes (*cf.* §3.7.2), démontre en effet que la source de polarisation interne borne l'évolution du courant de sortie de la cellule S2I à $-I_{BIAS}$ pour un courant d'entrée négatif (le courant sort de la cellule).

Considérons l'interconnexion de deux cellules $S2I_{w1}$ et $S2I_{w2}$:



Le fait de connecter deux cellules ensemble entraîne la même limitation sur la seconde cellule : le courant de sortie de la cellule S2I est limité à $-I_{BIAS}$:

- Pour I_{i1} négatif : $I_{i1\,min} = -I_{BIAS}$, soit : $I_{o1\,min} = -I_{BIAS}$
  , le courant I_{i2} est positif : $I_{i2\,max} = +I_{BIAS}$, soit : $I_{o2\,max} = +I_{BIAS}$

Dans le cas d'un courant d'entrée positif sur la première cellule, la seconde cellule provoquera à son tour la même limitation (son courant d'entrée sera négatif) : le courant de sortie de la seconde cellule sera à son tour bloqué à $+I_{BIAS}$.

- Pour I_{i1} positif, le courant I_{i2} est négatif : $I_{i2\,min} = -I_{BIAS}$, soit : $I_{o2\,min} = -I_{BIAS}$

La dynamique en courant d'un dispositif composé de deux cellules S2I interconnectées est par conséquent bornée à $\pm I_{BIAS}$.

Cette simulation permet de fixer une première spécification fonctionnelle sur la valeur du courant de polarisation. En se basant sur la valeur arbitraire de 150µA, nous garantissons une erreur relative de recopie entre courant d'entrée et de sortie maximale de l'ordre de 5% (voir pages annexes – résultats des mesures des performances de recopie de la cellule). Une nouvelle simulation intégrant ce paramètre pourra alors permettre de mesurer l'impact d'une telle erreur sur l'évolution des performances d'apprentissage du réseau.

Simulation avec le modèle comportemental de niveau 2 de la cellule S2I

Le modèle comportemental de niveau 2 de la cellule S2I est maintenant utilisé. Ce modèle introduit la source de polarisation interne aux cellules S2I à la valeur I_{BIAS} = 150µA. Une opération de normalisation des courants synaptiques est donc nécessaire pour adapter les fortes valeurs des courants issus de l'algorithme aux valeurs bornées par la source I_{BIAS} dans le dispositif de mémorisation CR-S2I. Pour des raisons de lisibilité, nous maintiendrons néanmoins la représentation des poids calculés par l'algorithme sur chacune des figures.
L'évolution des paramètres du réseau est donnée sur la figure 3-24 et le réseau converge après 81 cycles.

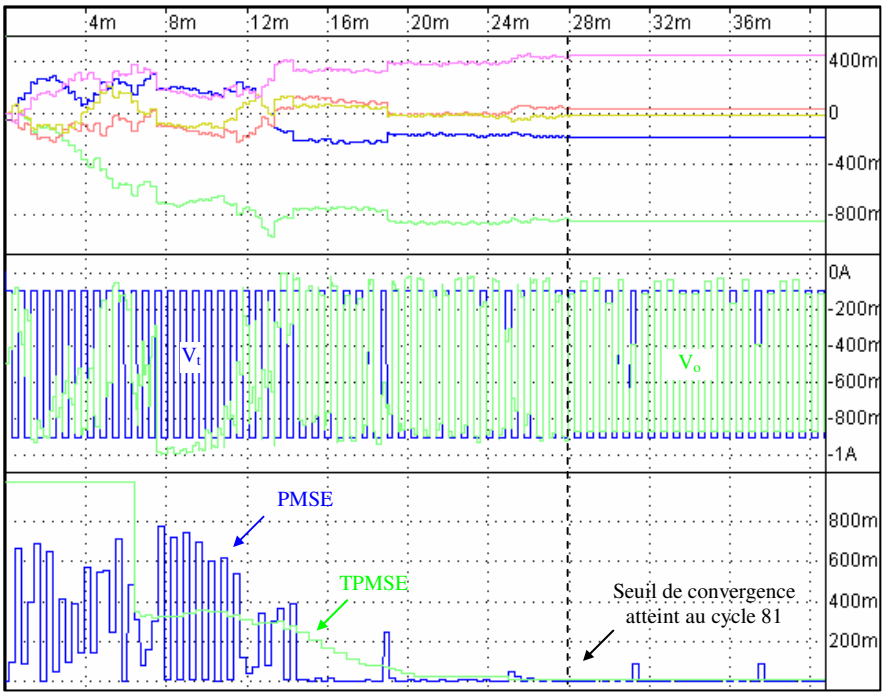


Figure 3-24 Simulation du modèle comportemental du réseau avec dispositif CR-S2I à cellules S2I comportemental de niveau 2

Simulation avec le modèle comportemental de niveau 3 de la cellule S2I

La simulation suivante emploie le modèle comportemental de niveau 3 de la cellule S2I où une résistance interne est ajoutée à la source de courant de polarisation. Les évolutions des paramètres du réseau sont représentées sur la figure 3-25 et le réseau converge en 142 cycles pour une valeur de résistance interne R_{BIAS}=1MΩ. Une valeur de R_{BIAS} inférieure ou supérieure dégrade les performances d'apprentissage. Le tableau reprend les différents résultats obtenus pour trois valeurs différentes de R_{BIAS} :

Performances d'apprentissage pour le modèle comportemental de niveau 3			
R_{BIAS}	100kΩ	1 MΩ	10 MΩ
Cycles d'apprentissage	Pas de convergence	142	165

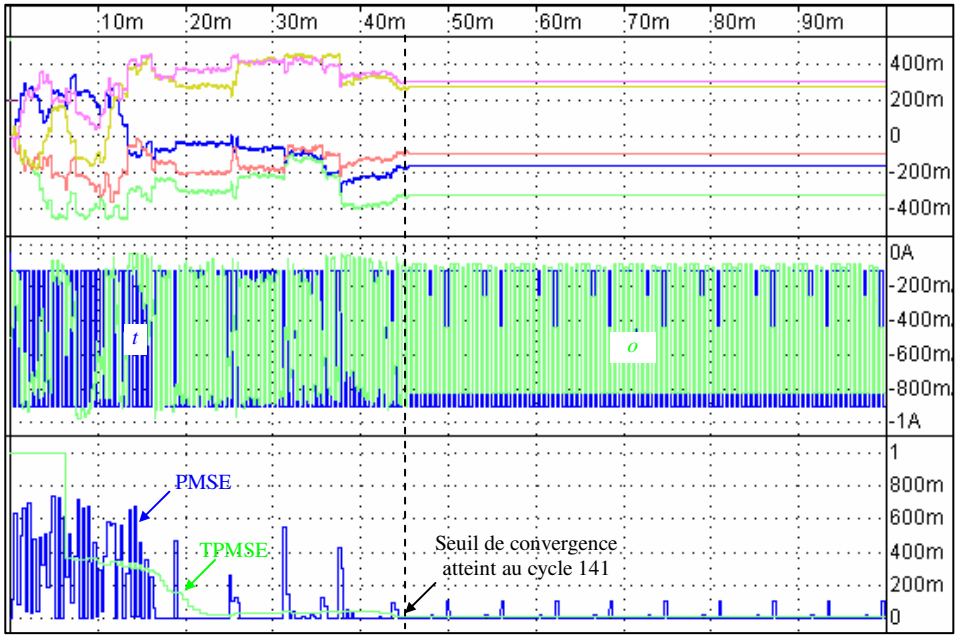


Figure 3-25 Simulation du modèle comportemental du réseau avec dispositif CR-S2I à cellules S2I comportemental de niveau 3(R_{BIAS} =1MΩ)

Notons également que la résistance de sortie de la cellule a été fixée à R_o=100MΩ. Des valeurs inférieures limitent les chances de convergence de l'algorithme en phase d'apprentissage.

A ce stade de simulations, nous pouvons poser une première base des paramètres de dimensionnement du réseau :

Proposition de dimensionnement du modèle comportemental de niveau 3		
R_o	I_{BIAS}	R_{BIAS}
100MΩ	150µA	10MΩ

Cette proposition de dimensionnement nous permet de définir les valeurs du modèle structurel de niveau 1 de la cellule.

Simulation avec le modèle structurel de niveau 1 de la cellule S2I

La simulation de l'application avec le modèle structurel de niveau 1 de la cellule S2I est donnée sur la figure 3-26. Le processus d'apprentissage converge après 274 cycles.

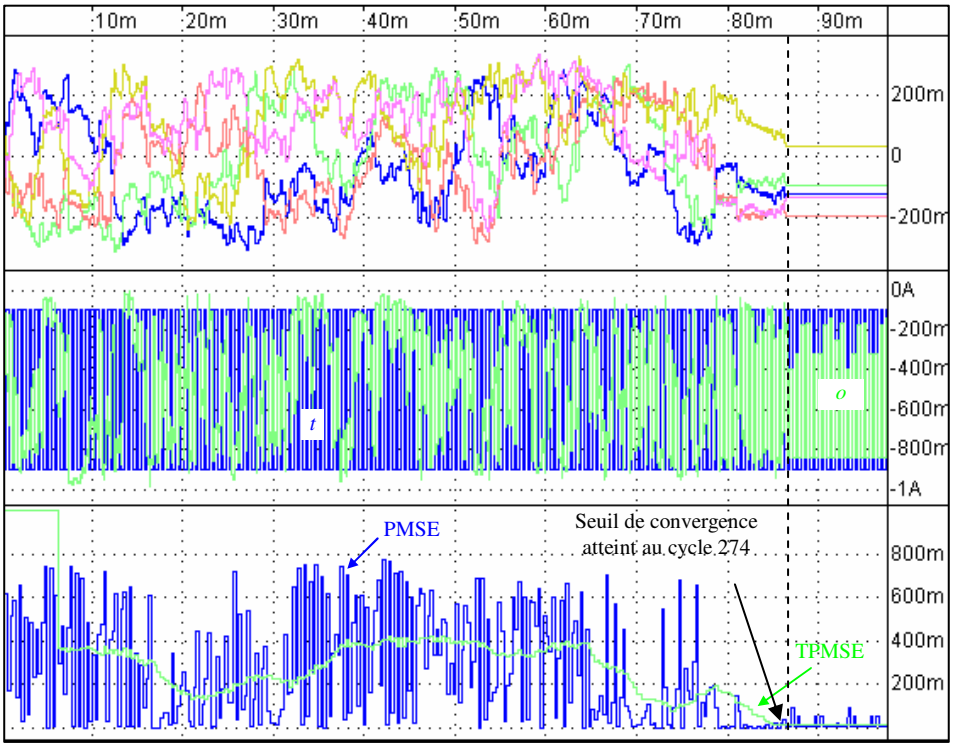


Figure 3-26 Simulation du modèle comportemental du réseau avec dispositif CR-S2I à cellules S2I structurel de niveau 1

3.5.2.3 Evaluation du dispositif et perspectives

Le réseau a été simulé en introduisant les modèles de premier ordre de composants (transistors en particulier). Les simulations ont permis d'évaluer la faisabilité des choix architecturaux d'un dispositif CR-S2I dans le cadre de notre application (réseau dédié à la reconnaissance de caractères chiffrés). Le régime spécifique de fonctionnement de ce type de dispositif fait intervenir un régime de commutation de courants. La caractérisation d'un tel dispositif nécessite d'imposer un état continu de fonctionnement et une analyse DC a été par conséquent effectuée.

Le dispositif CR-S2I d'une branche synaptique requiert quatre cellules S2I, et chacune d'elles fonctionne avec une source de courant de 150µA sous une tension d'alimentation de 2,5V. La consommation en courant par dispositif s'élève par conséquent à 600µA soit un total de 21.6mA sur l'ensemble du réseau.

Le réseau de neurones est composé de blocs supplémentaires (blocs à transconductance, blocs à synapses, bloc à neurone) dont le régime de fonctionnement est continu. La fonction de transfert de ces blocs permet de mettre en relation le dimensionnement des composants, les contraintes de fonctionnement et les spécifications à atteindre. Dimensionner le réseau et ses blocs nécessite d'adopter une démarche globale de la conception en tenant compte notamment de l'aspect hétérogène des contraintes. En effet, même s'il est possible, localement, de garantir le

dimensionnement optimal des composantes d'un système, rien ne garantit que le système considéré dans sa globalité possède des performances optimales. La présence d'interactions entre les différents sous-systèmes couple fortement les éléments entre eux : on peut relever par exemple la dépendance entre les caractéristiques intrinsèques des composants d'un circuit et son environnement.

Comment, dans ce contexte, assurer le comportement optimal de cet ensemble d'éléments en interaction ?

Nous proposons alors de porter un regard un peu plus distant sur les techniques de dimensionnement actuelles et voir selon quelles approches est traitée la conception de systèmes. Ce tour d'horizon sera abordé dans le prochain chapitre.

3.6 Conclusion

Dans ce chapitre, nous avons fait le choix du langage VHDL-AMS pour décrire le réseau de neurones. A partir de ce langage, un modèle comportemental du réseau et de son apprentissage a pu être proposé et validé par comparaison avec le modèle conceptuel élaboré sous MATLAB. Une architecture fonctionnelle du réseau a été ensuite présentée à partir de laquelle un modèle comportemental puis structurel d'une branche synaptique du réseau a été simulé. Ce modèle est bâti autour de cellules mémoire à courant commuté dont le rôle est de maintenir le courant des poids synaptiques à un niveau constant le temps d'un cycle d'apprentissage. Les modèles des composants de cette cellule ont été tour à tour modélisés selon une démarche Top-down c'est-à-dire à partir d'un haut niveau d'abstraction vers un niveau plus détaillé. Le comportement dynamique du réseau et de son algorithme d'apprentissage ont été simulés dans le cadre de la reconnaissance de caractères chiffrés et une première proposition de dimensionnement des composants des cellules mémoire à courant commuté a pu être proposée.
Ce travail de dimensionnement permet en première approche de valider les concepts et les choix architecturaux de l'application en se basant uniquement sur l'aspect des performances d'apprentissage. Mais, si l'on ajoute à cette considération la recherche de performances particulières (gain d'un OTA, gestion de l'énergie,…) tout en assurant la mission confiée à l'application, la conception devient quasiment impossible sans outil adapté. Avant de poursuivre ce travail, il parait utile d'avoir une vision globale des méthodologies actuelles d'aide à la conception. Aussi, le prochain chapitre propose de dresser un bilan des méthodes et techniques de conception en électronique.

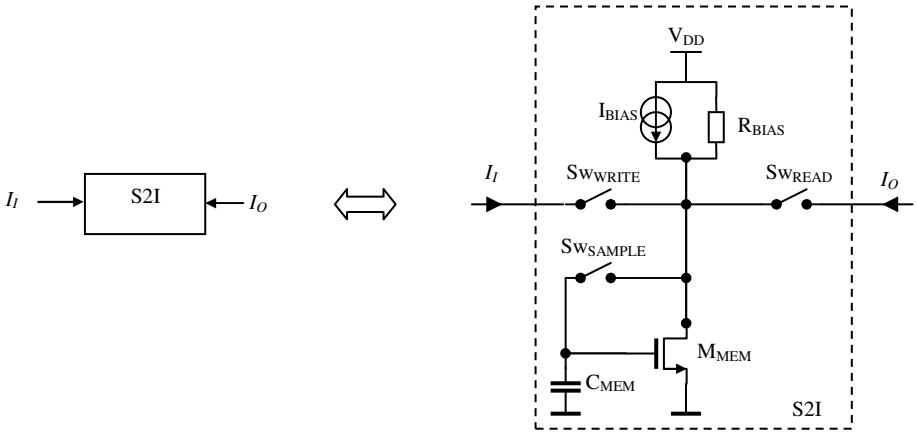
3.7 Annexe du chapitre 3

La partie annexe comprend :

- la description du modèle de la cellule à courant commuté S2I ;
- les résultats des simulations pour évaluer la plage de fonctionnement de la cellule ;
- les résultats des mesures des performances de recopie de la cellule ;
- le code VHDL-AMS des modèles de la cellule S2I.

3.7.1 Description du modèle de la cellule à courant commuté S2I

Le schéma de la cellule à courant commuté est donné ci-dessous (les signaux de commande des commutateurs ne sont pas ici représentés) :



Le modèle VHDL-AMS de la cellule est dicté par les lois physiques simples. Ainsi, toute influence liée à la géométrie du composant ou à sa technologie est dans négligée ce premier modèle.

La source de courant de polarisation du transistor possède une résistance interne R_{BIAS}.
L'effet de modulation de la longueur du canal du transistor MOS est une fonction linéaire et son courant de drain en régime de saturation a pour expression :

$$I_D = \frac{1}{2} K \frac{W}{L} (V_{GS} - V_{TH})^2 (1 + \lambda V_{DS}) \quad \text{pour : } V_{DS} \geq V_{DSsat} = V_{GS} - V_{TH}$$

En régime linéaire :

$$I_D = K \quad (V_{GS} - V_{TH} - \frac{V_{DS}}{2}) V_{DS} (1 + \lambda V_{DS}) \quad \text{pour : } o \leq V_{DS} \leq V_{DSsat}$$

Sinon :

$$I_D = o$$

Les paramètres utilisés pour les simulations sont les suivants :

V_{To} [mV]	μ_o [$cm^2V^{-1}s^{-1}$]	p_{sub} [cm^{-3}]	C_{OX} [F/cm^2]	W [µm]	L [µm]
535	285	$6{,}59.10^{17}$	$0.5.10^{-6}$	10.0	10.0

Tension de seuil : $V_{TH} = V_{To} + \gamma (\sqrt{\Phi - V_{BS}} - \sqrt{\Phi})$ où : $\gamma = \sqrt{\frac{2q\varepsilon_{si}}{C_{OX}}} = 1.15.10^{-13}$

3.7.2 Analyse fonctionnelle de la cellule S2I

On souhaite évaluer le domaine de fonctionnement du modèle structurel d'une cellule S2I. Le domaine de fonctionnement de la cellule est le domaine pour lequel la cellule est capable de garantir la recopie du courant d'entrée sur sa sortie. La précision de recopie de la cellule est évaluée par une mesure de la différence des courants entrants et sortants : cette mesure est réalisée en simulation DC à partir du dispositif de caractérisation donné sur la figure 3-27 où deux cellules sont interconnectées : le courant de sortie de la première cellule sera le courant d'entrée de la seconde cellule. Les capacités de mémorisation C_{MEM} ont donc été supprimées.

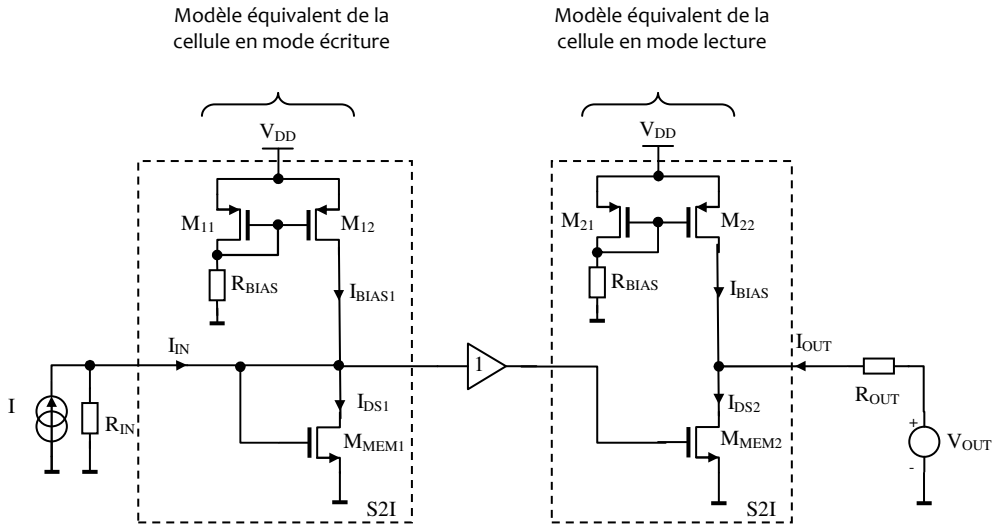


Figure 3-27 Dispositif de test de la cellule S2I (mode échantillonnage et lecture)

Le courant d'entrée I est injecté dans la cellule en mode écriture (courant I_{IN}). Ce courant fixe la tension de drain V_{DS} du transistor de mémorisation. La tension de drain est alors transmise via un suiveur idéal (impédance d'entrée infinie) à la cellule en mode lecture. Le courant de sortie de la cellule est donc le courant absorbé ou débité par la source de tension de sortie V_{OUT} d'impédance R_{OUT}.

Les valeurs des composants du circuit :

I_{BIAS} [µA]	R_{IN} [Ω]	C_{MEM} [pF]	V_{OUT} [V]	R_{OUT} [Ω]	R_{BIAS} [Ω]	V_{DD} [V]
10.0	1.10^6	5.0	1.0	1.0	1.10^5	2.0

Transistors M_{MEMi}, M_{i1} et M_{i2} :

V_{To} [mV]	μ_o [$cm^2V^{-1}s^{-1}$]	p_{sub} n_{sub} [cm^{-3}]	C_{OX} [F/cm^2]	W [µm]	L [µm]
535	285	$6{,}59.10^{17}$	$0.5.10^{-6}$	10.0	10.0

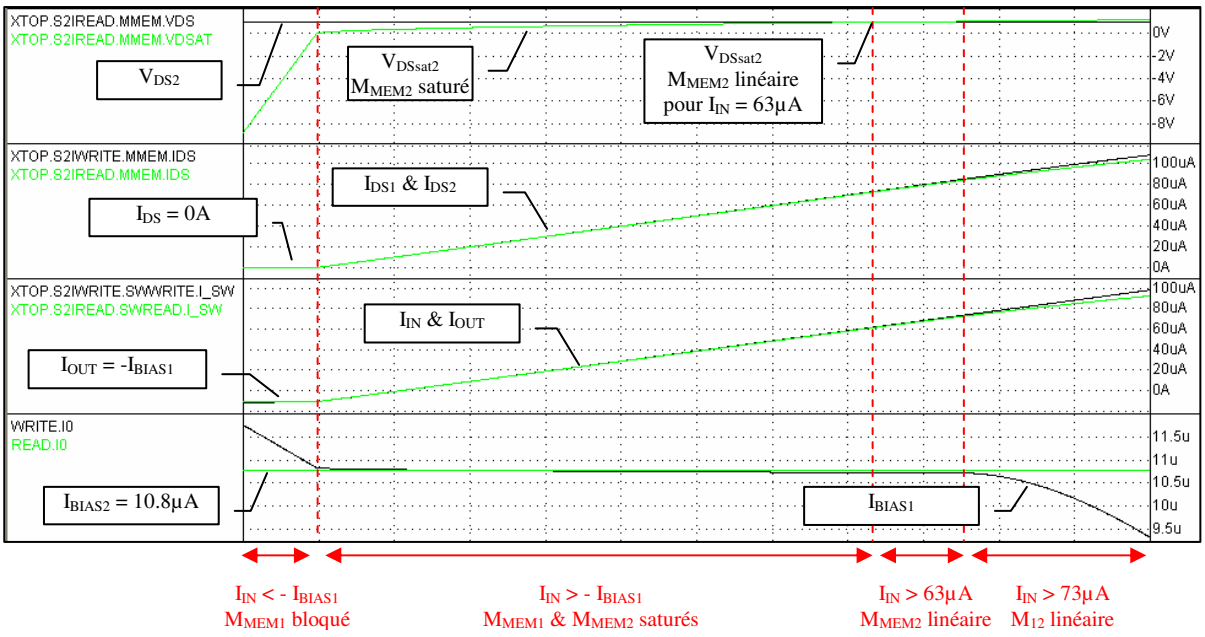
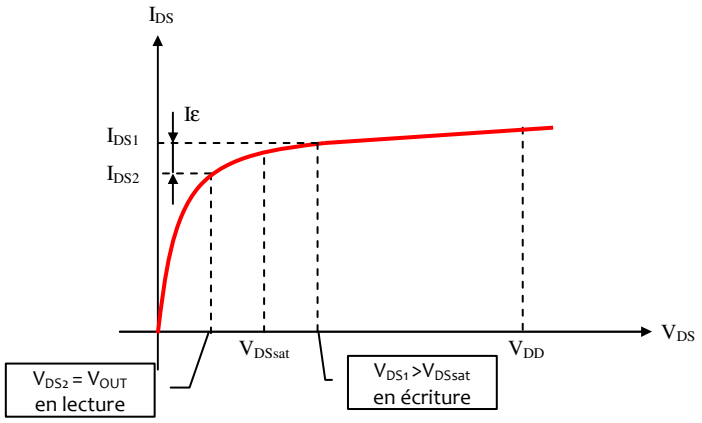


Figure 3-28 Cellule S2I en mode lecture et écriture avec source de courant à miroir de courant

La figure 3-28 donne l'ensemble des relevés des cellules équivalentes en mode échantillonnage et lecture. Dans cette simulation DC, le courant d'entrée I_{IN} évolue en rampe dans la plage [-20.0µA ; +100.0µA]. En mode écriture et pour un courant d'entrée I_{IN} négatif et supérieur en valeur absolue au courant I_{BIAS1}, le transistor M_{MEM1} est bloqué (I_{DS}=0.0) avec une tension V_{DS1} fortement négative : la cellule n'est donc pas capable de mémoriser un courant d'entrée pour ces valeurs.
Pour les valeurs de I_{IN} croissantes et supérieures à $-I_{BIAS1}$, le transistor de mémorisation M_{MEM1} est en régime de saturation jusqu'à la valeur de I_{IN}=63.0µA et le courant d'entrée en mode écriture est restitué en sortie de la cellule en mode lecture. Au-delà de cette valeur, le transistor passe en régime linéaire en mode lecture : la tension V_{DS} reste fixée à V_{OUT} alors que la tension V_{GS} (et donc V_{DSsat}) déterminée en régime d'écriture continue à augmenter :

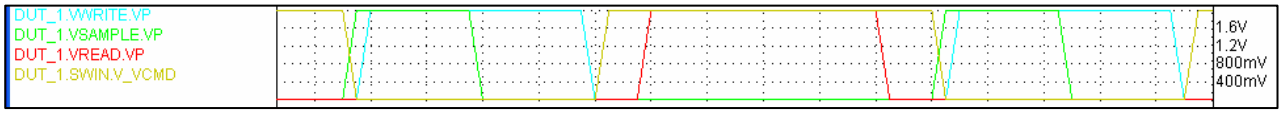


Le courant de sortie I_{OUT} est donc limité par le courant I_{DS2} du transistor M_{MEM2}, on relève donc à partir de I_{IN}=63.0µA un écart entre le courant entrant et sortant de la cellule qui tend à s'accentuer à mesure que I_{IN} augmente. A partir de la valeur particulière I_{IN}=73.0µA, la tension V_{DS}

fait rentrer le transistor M_{12} de la source de courant en régime linéaire : le courant I_{BIAS1} commence à chuter.

3.7.3 Mesure des performances de recopie de la cellule

On souhaite mesurer les performances de recopie de la cellule. La cellule est pilotée en mode écriture et lecture. L'état de mémorisation sépare les deux modes (temps morts). Les signaux de commandes sont les suivants :



Le commutateur Sw_{SAMPLE} précède Sw_{WRITE} dans sa fermeture. C'est le cycle d'échantillonnage. Le commutateur Sw_{WRITE} se ferme ensuite (Sw_{SAMPLE} est ouvert) : c'est le cycle d'écriture. Après une période de mémorisation (Sw_{SAMPLE} et Sw_{WRITE} ouverts), le cycle de lecture débute avec la fermeture de Sw_{READ}.

Une expérience permet d'évaluer la précision de recopie du modèle structurel de niveau 1 de la cellule S2I. Le courant d'entrée Ii de la cellule décroît à partir de la valeur 150µA. Les résultats sont donnés sur la figure 3-29. L'erreur relative calculée par le rapport des courants sortant et entrant Io/Ii reste inférieure à 5% sur l'ensemble de la plage. Elle devient même inférieure à 1% pour un courant d'entrée passant sous 5µA.

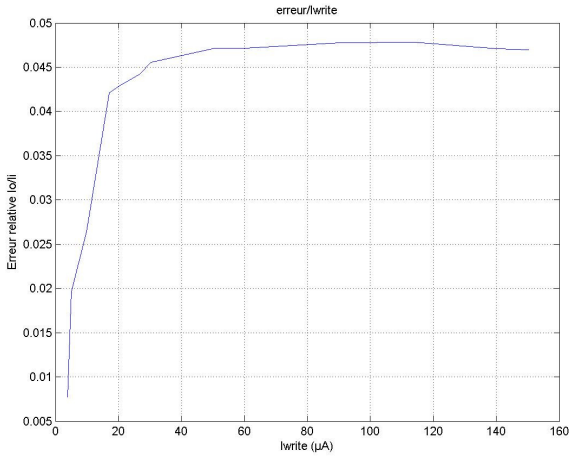


Figure 3-29 Erreur relative Io/Ii du modèle structurel de niveau 1 de la cellule S2I

3.7.4 Code VHDL-AMS des modèles de la cellule S2I

```
ENTITY S2I IS
    GENERIC(Ib : Real);
    PORT(TERMINAL T_in, T_out, T_write, T_sample, T_read, T_VDD, T_VSS :
electrical);
END ENTITY;

-- **** behavioral model level 1 ****

ARCHITECTURE bhv1 OF S2I IS
    CONSTANT Ri : real := 1.0e0;
    CONSTANT Ro : real := 1.0e8;
    CONSTANT Vth : real := 1.0;
    CONSTANT tr, tf : time := 1 us;
    TERMINAL T_i, T_o : electrical;
    QUANTITY V_i ACROSS I_i THROUGH T_i;
```

```
    QUANTITY V_o ACROSS I_o THROUGH T_o;
    QUANTITY V_write ACROSS I_write THROUGH T_write;
    QUANTITY V_sample ACROSS I_sample THROUGH T_sample;
    QUANTITY V_read ACROSS I_read THROUGH T_read;
    SIGNAL S_id : real := 0.0;
    FUNCTION t2r(t : time) RETURN real IS
    BEGIN
        RETURN real(t/1fs)*1.0e-15;
    END FUNCTION;
BEGIN
    V_i == Ri*I_i;
    I_o == S_id'RAMP(t2r(tr), t2r(tf)) + V_o/Ro;
    V_write == Ri*I_write;
    V_sample == Ri*I_sample;
    V_read == Ri*I_read;
    PROCESS
        VARIABLE id : real := 0.0;
    BEGIN
        WAIT UNTIL NOT V_sample'ABOVE(Vth);
        id := I_i;
        WAIT UNTIL V_read'ABOVE(Vth);
        S_id <= id;
        WAIT UNTIL NOT V_read'ABOVE(Vth);
        S_id <= 0.0;
    END PROCESS;

    SwWrite:    ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_in, T_cmd=>T_write, T_2=>T_i);
    SwRead:     ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_out, T_cmd=>T_read, T_2=>T_o);
END ARCHITECTURE;

-- **** behavioral model level 2 ****

ARCHITECTURE bhv2 OF S2I IS
    CONSTANT Ri : real := 1.0e0;
    CONSTANT Ro : real := 1.0e8;
    CONSTANT Vth : real := 1.0;
    CONSTANT tr, tf : time := 1 us;
    TERMINAL T_i, T_o : electrical;
    QUANTITY V_i ACROSS I_i THROUGH T_i;
    QUANTITY V_o ACROSS I_o THROUGH T_o;
    QUANTITY V_write ACROSS I_write THROUGH T_write;
    QUANTITY V_sample ACROSS I_sample THROUGH T_sample;
    QUANTITY V_read ACROSS I_read THROUGH T_read;
    SIGNAL S_id : real := 0.0;
    FUNCTION t2r(t : time) RETURN real IS
    BEGIN
        RETURN real(t/1fs)*1.0e-15;
    END FUNCTION;
BEGIN
    V_i == Ri*(I_i - Ib); -- bias current <0
    I_o == S_id'RAMP(t2r(tr), t2r(tf)) + V_o/Ro + Ib;
    V_write == Ri*I_write;
    V_sample == Ri*I_sample;
    V_read == Ri*I_read;
    PROCESS
        VARIABLE id : real := 0.0;
    BEGIN
        WAIT UNTIL NOT V_sample'ABOVE(Vth);
        id := I_i - Ib;
        WAIT UNTIL V_read'ABOVE(Vth);
        IF id<0.0 THEN
            S_id <= 0.0; -- MOS is turn off
```

```
        ELSE
            S_id <= id;
        END IF;
        WAIT UNTIL NOT V_read'ABOVE(Vth);
        S_id <= 0.0;
    END PROCESS;

    SwWrite:    ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_in, T_cmd=>T_write, T_2=>T_i);
    SwRead:     ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_out, T_cmd=>T_read, T_2=>T_o);
END ARCHITECTURE;

-- **** behavioral model level 3 ****

ARCHITECTURE bhv3 OF S2I IS
    CONSTANT Ri : real := 1.0e0;
    CONSTANT Ro : real := 1.0e8;
    CONSTANT Rb : real := 1.0e6;
    CONSTANT Vth : real := 1.0;
    CONSTANT Vsup : real := 2.5;
    CONSTANT tr, tf : time := 1 us;
    TERMINAL T_i, T_o : electrical;
    QUANTITY V_i ACROSS I_i THROUGH T_i;
    QUANTITY V_o ACROSS I_o THROUGH T_o;
    QUANTITY V_write ACROSS I_write THROUGH T_write;
    QUANTITY V_sample ACROSS I_sample THROUGH T_sample;
    QUANTITY V_read ACROSS I_read THROUGH T_read;
    SIGNAL S_id : real := 0.0;
    FUNCTION t2r(t : time) RETURN real IS
    BEGIN
        RETURN real(t/1fs)*1.0e-15;
    END FUNCTION;
BEGIN
    V_i == Ri*(I_i - Ib + (Vsup - V_o)/Rb); -- bias current <0
    I_o == S_id'RAMP(t2r(tr), t2r(tf)) + V_o/Ro + Ib - (Vsup - V_o)/Rb;
    V_write == Ri*I_write;
    V_sample == Ri*I_sample;
    V_read == Ri*I_read;
    PROCESS
        VARIABLE id : real := 0.0;
    BEGIN
        WAIT UNTIL NOT V_sample'ABOVE(Vth);
        id := I_i - Ib;
        WAIT UNTIL V_read'ABOVE(Vth);
        IF id<0.0 THEN
            S_id <= 0.0; -- MOS is turn off
        ELSE
            S_id <= id;
        END IF;
        WAIT UNTIL NOT V_read'ABOVE(Vth);
        S_id <= 0.0;
    END PROCESS;

    SwWrite:    ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_in, T_cmd=>T_write, T_2=>T_i);
    SwRead:     ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_out, T_cmd=>T_read, T_2=>T_o);
END ARCHITECTURE;

-- **** strutural model level 1 ****
```

```
ARCHITECTURE str1 OF S2I IS
    CONSTANT Vth : real := 1.0;
    TERMINAL T_D, T_G, T_IREF : electrical;
BEGIN
    Ibias:      ENTITY IG(bhv)
                GENERIC MAP(a=>0.0, f=>0.0, s=>0.0, o=>Ib)
                PORT MAP(T_p=>T_D, T_m=>T_VDD);
    Rbias :     ENTITY Res(bhv)
                GENERIC MAP(R=>1.0e5)
                PORT MAP(T_1=>T_VDD, T_2=>T_D);
    Mmem :      ENTITY MOS(bhv2)
                GENERIC MAP(p=>+1.0,vt0=>535.0e-
3,u0=>285.0,nsub=>6.59e17,cox=>0.5e-6,w=>10.0e-6,l=>10.0e-6,tk=>300.0)
                PORT MAP(T_D=>T_D, T_G=>T_G, T_S=>ground, T_B=>ground);
    Cds :       ENTITY Cap(bhv)
                GENERIC MAP(C=>1.0e-12)
                PORT MAP(T_1=>T_D, T_2=>ground);
    Cmem:       ENTITY Cap(bhv)
                GENERIC MAP(C=>5.0e-12)
                PORT MAP(T_1=>T_G, T_2=>ground);
    SwWrite:    ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_in, T_cmd=>T_write, T_2=>T_D);
    SwSample:   ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_D, T_cmd=>T_sample, T_2=>T_G);
    SwRead:     ENTITY Sw(bhv)
                GENERIC MAP(th=>Vth)
                PORT MAP(T_1=>T_out, T_cmd=>T_read, T_2=>T_D);
END ARCHITECTURE;
```

Chapitre 4 - le dimensionnement d'une architecture par l'exploration de l'espace de conception

4.1 Introduction

Les deux premiers chapitres de cette thèse ont permis de dégager une première architecture fonctionnelle du réseau de neurones. Une attention a été portée spécifiquement sur la réalisation du dispositif de mémorisation des courants synaptiques et un premier schéma structurel a été proposé à partir de cellules S2I. Néanmoins, de telles cellules reposent sur un régime de fonctionnement à courants commutés rendant leur dimensionnement complexe. Les méthodes de dimensionnement que nous envisageons d'utiliser ne permettent pas actuellement de traiter ce type de problème et le dimensionnement d'une cellule S2I est typiquement fait manuellement. En revanche, un circuit tel qu'un OTA ou un multiplieur fonctionne dans un régime continu (non commuté) c'est-à-dire qu'il existe une relation entre les grandeurs d'entrée et de sortie qui est valide à tout instant *t*. C'est à travers cette relation que nous souhaitons poser notre démarche de dimensionnement. Cette démarche sera alors l'occasion d'essayer de développer des liens entre les différents niveaux hiérarchiques qui interviennent en phase de conception.

Mais dimensionner un circuit en régime continu induit de nouvelles problématiques.

Un circuit analogique repose sur un jeu d'équations caractérisant son fonctionnement. Dimensionner un circuit analogique nécessite une prise en compte d'objectifs fonctionnels et de contraintes physiques. Le circuit doit être en effet dimensionné en fonction d'objectifs de performances (gain, bande-passante, …) à atteindre mais aussi en fonction de contraintes de fonctionnement (dynamique de signal, régime de fonctionnement, …), ou de fabrication (surface de silicium, limites dissipatives, limites technologiques, …). Le dimensionnement de circuits met en jeu des équations devenues complexes et rechercher la solution satisfaisant ces équations nécessite des outils adaptés. Au niveau système, il devient difficile de déterminer comment le dimensionnement d'un sous-système va influer sur les autres, et si les améliorations locales de certaines performances ne se font pas au détriment des autres voir du système dans son ensemble.

Dans ce contexte, les méthodes traditionnelles de conception mettent en œuvre des processus d'optimisation qui visent à satisfaire le plus grand nombre d'objectifs tout en respectant un grand nombre de contraintes. Le dimensionnement d'un circuit analogique s'exprime alors naturellement comme un problème d'optimisation multi-objectif sous contraintes.

Mais résoudre un tel problème nécessite de satisfaire toutes les contraintes. Or, il apparaît qu'un circuit analogique possède typiquement un jeu d'équations sous-contraint c'est-à-dire avec trop de degrés de liberté [Gie-05a, Rut-07]. Déterminer les valeurs des variables « libres » d'un tel jeu d'équations est alors plus délicat. En cas d'échec, le processus d'optimisation est alors réitéré sans garantir qu'il convergera. La recherche d'une solution peut donc être longue et incertaine.
Avec l'accroissement de la complexité des systèmes à concevoir, la pression du « time to market », et l'importance d'une bonne robustesse des circuits aux dérives de fabrication, le développement de nouveaux outils de conception devient une réelle urgence [Gie-05b].
La tendance qui se dégage est alors de mieux préparer la phase d'optimisation en offrant des conditions initiales afin d'augmenter les chances de convergence. Une réponse possible est alors de rechercher prioritairement le sous-espace des variables qui satisfait toutes les contraintes du problème. Ce premier problème est un CSP (Constraint Satisfaction Problem). Notre approche vise ainsi à centrer l'exploration de l'espace de conception au voisinage du point optimal. Nous espérons alors d'une part prouver rapidement l'existence de ce point et d'autre part lui garantir un minimum de robustesse.

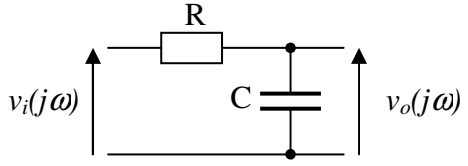
En vue d'analyser les besoins, ce chapitre propose une synthèse des processus de conception d'un circuit analogique. Elle permettra ainsi de mieux identifier les obstacles actuels qui ralentissent le processus de conception et les liens possibles avec les différents niveaux dans la hiérarchie. De nouvelles pistes d'aide à la conception seront ensuite présentées.
Le chapitre est organisé de la façon suivante. La partie 2 pose la terminologie nécessaire au dimensionnement d'une architecture. La partie 3 décrit le formalisme d'un processus d'optimisation. La partie 4 présente les approches d'optimisation où leur mise en œuvre pratique dans les outils d'aide à la conception qui sont alors comparées. La partie 5 se focalise sur l'approche d'optimisation par équations et en présente les avantages. La partie 6 donnera les fondements de notre approche de conception.

4.2 Formalisation d'un problème de dimensionnement d'une architecture

Avant d'aborder plus en détail les approches spécifiques utilisées lors du dimensionnement d'une architecture, cette partie donne la démarche générale du concepteur de circuit intégré et le vocabulaire associé.

Dans le dimensionnement d'un circuit, interviennent les attentes de l'utilisateur et les lois spécifiques en électricité (les lois de Kirchhoff). L'utilisateur du système exprime ses attentes fonctionnelles et ses attentes de qualité de fonctionnement. Il devient donc nécessaire de traduire ces attentes en objectifs de conception. Les lois de Kirchhoff expriment à leur tour l'équilibre des grandeurs électriques à tout instant. Elles forment donc la condition fondamentale à satisfaire.
L'aspect fonctionnel vient du comportement relatif d'un signal par rapport à un autre signal du circuit. L'utilisateur fait le choix d'un signal de sortie dont le comportement doit être en relation avec une entrée de référence. Cette relation traduit la fonctionnalité du circuit à travers la notion de fonction de transfert lorsque l'hypothèse de régime harmonique est retenue. Une telle hypothèse permet classiquement de caractériser le circuit et donc de le dimensionner.

Considérons par exemple le cas d'un filtre passe-bas du premier ordre. Pour dimensionner les composants de ce filtre, nous exprimons sa fonction de transfert :



Fonction de transfert complexe du filtre :

$$\overline{T(j\omega)} = \frac{\overline{v_o}(j\omega)}{\overline{v_i}(j\omega)} = \frac{1}{1+jRC\omega} = \underbrace{|\overline{T(j\omega)}|}_{gain}.e^{\overbrace{j\,Arg(T(j\omega))}^{phase}}$$

La fonction de transfert fait alors apparaître les caractéristiques de gain et de phase du filtre dépendantes de degrés de libertés que sont les paramètres R, C et de la variable libre ω. La nature des composants est donc implicitement prise en compte dans la fonction de transfert. L'état d'équilibre des grandeurs électriques à un instant *t* dépend par conséquent de la topologie du circuit et de ses composants.

Les attentes fonctionnelles du concepteur reposent ici sur l'expression des grandeurs de gain et de phase du filtre :

Expression du gain :

$$|\overline{T(j\omega)}| = \left|\frac{\overline{v_o}(j\omega)}{\overline{v_i}(j\omega)}\right| = \frac{1}{\sqrt{1+(RC\omega)^2}}$$

Expression de la phase :

$$Arg(\overline{T(j\omega)}) = -Arctg(RC\omega)$$

Ces grandeurs traduisent les performances du filtre c'est-à-dire sa fonctionnalité. Ces performances doivent atteindre des objectifs de conception que l'on désignera par les spécifications. Le concepteur devra donc déterminer, pour une plage de ω, les valeurs des composants R et C afin de faire correspondre les performances du filtre à un gabarit de spécifications en gain et en phase.

En généralisant, un circuit possède plusieurs degrés de libertés caractérisés par les grandeurs électriques (amplitude, fréquence, …) et les grandeurs de composants (résistances, capacités, …). Elles représentent les variables d'ajustement du circuit. A partir de ces variables d'ajustement, on exprime la fonction de transfert qui caractérise les performances du circuit (gain, phase, …). Parmi les attentes de l'utilisateur apparaissent également les objectifs à donner pour chacune de ces performances par rapport à une plage d'acceptation ; ceci revient à formuler des spécifications qui correspondent à une plage bornée de valeurs.

Pour résumer nous trouvons trois grandeurs qui interviennent dans le dimensionnement d'un circuit :

- *Les ajustements* : Les variables qui correspondent aux valeurs de composants ou encore des états électriques (courant de polarisation, …) sont appelées les « ajustements ». Nous désignerons par la lettre **A** l'espace de ces ajustements et *a* une valeur particulière de cet espace.

- *Les performances* : Les résultats en terme de caractéristiques de fonctionnement (gain, bande passante, …) sont appelés les « performances » obtenues par le circuit. Nous les désignerons par la lettre **P** l'espace de ces performances et *p* une valeur particulière de cet espace.

- *Les spécifications* : Les objectifs imposés par l'utilisateur aux performances que l'on souhaite contrôler avec une certaine tolérance seront appelés les « spécifications ». Nous les désignerons par la lettre **S**. Une spécification est une restriction du domaine des performances. **S** est par conséquent un sous-espace de **P**.

Résoudre un problème de dimensionnement consiste à déterminer toutes les variables d'ajustements qui fixent les performances du circuit relativement aux spécifications. Le problème peut être reformulé de la façon suivante : en partant des spécifications et d'un ensemble de performances, comment en déduire les valeurs des variables d'ajustement ? Ce problème s'exprime alors comme un problème inverse.

Le dimensionnement d'un circuit est associé à une application *f* de l'espace des ajustements dans l'espace des performances que nous avons présenté comme la fonction de transfert. L'objectif du dimensionnement est de déterminer un point *a* de l'espace **A** pour lequel un point *p* des performances appartient au sous-espace des spécifications *S*. Par conséquent, le sous-espace des solutions $A_S \in$ **A** est un sous-espace de **A** qui vérifie :

$$A_S = \{\forall a \in A_S / p = f(a) \in S\}$$

Pour déterminer un point $a_S \in A_S$, un solveur d'équations est alors utilisé. Deux situations se dégagent et conditionnent le type de méthode de résolution numérique utilisé.

Si la fonction *f* est linéaire, sa fonction inverse est explicitable. La résolution numérique du jeu d'équations est alors directe. En choisissant un point $p_k = p_S$ inclus dans l'intervalle de

spécifications S = (ΔS_x,ΔS_y), on détermine directement le point $a_k = a_S \in A_S$. Ce cas est illustré sur la figure 4-1(a). Cette situation est néanmoins peu représentative des problèmes de dimensionnement rencontrés en pratique où la fonction *f* est le plus généralement non linéaire.

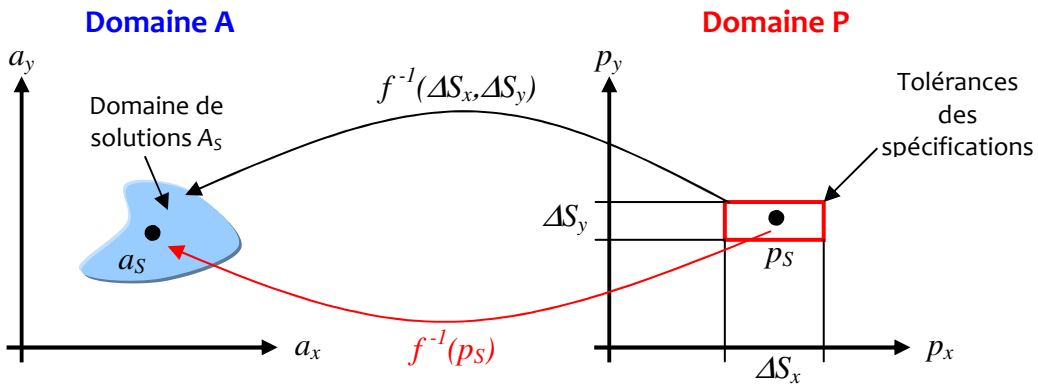


Figure 4-1(a)

Si la fonction *f* est non linéaire, sa fonction inverse n'est pas forcément explicitable. La résolution numérique du jeu d'équations est alors itérative. Cette situation est la situation la plus communément rencontrée dans les problèmes de dimensionnement actuels. Elle s'appuie sur une démarche qui peut se formaliser de la manière suivante : le concepteur choisit un point a_k et calcule son image p_k par la fonction *f*. Il vérifie ensuite l'appartenance du point p_k à l'intervalle de spécifications S = (ΔS_x,ΔS_y). Si le test $p_k \in S$ est vérifié alors $a_k = a_S \in A_S$. Dans le cas contraire, le point $a_k \notin A_S$, et un point $\boldsymbol{a_k}$ avec un nouvel indice *k* est choisi. La démarche est illustrée sur la figure 4-1(b) avec trois points (a_1, a_2, et a_3) pris dans l'espace **A** correspondant à trois points (p_1, p_2, et p_3) de l'espace **P**. Les points (a_1, a_2) « produisent » deux points (p_1, p_2) qui n'appartiennent pas à S. Ces deux points sont donc tour à tour rejetés. Finalement, seul le point a_3 permet de vérifier la condition $p_3 \in S$. Ce point sera donc un point de A_S.

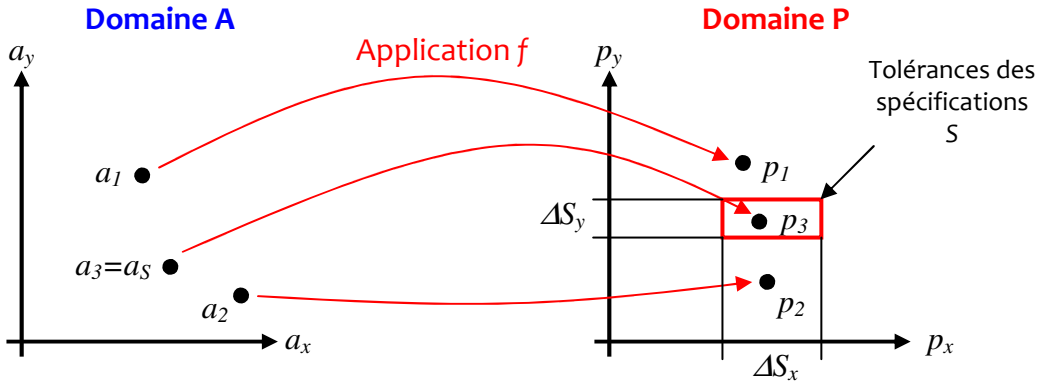


Figure 4-1(b)

Figure 4-1 Espace des ajustements lié à l'espace des performances par une application *f* sous les tolérances de spécifications

Cette démarche souligne toute la difficulté à déterminer l'espace solution A_S lorsque la relation *f* n'est pas explicitable. Or, dans les cas pratiques de dimensionnement les relations de performances font intervenir régulièrement des fonctions fortement non linéaires dont l'explicitation de la fonction inverse n'est pas garantie.

Dimensionner un circuit en résolvant un système d'équations non linéaires est aujourd'hui quasiment impossible sans l'apport des méthodes de l'analyse numérique notamment lorsque les critères de performances à atteindre sont multiples. Le recours à ces méthodes permet d'approcher les solutions d'un système avec un niveau de précision fixé par le calculateur qui implémente cette méthode. Le processus de dimensionnement devient un processus itératif qui retourne finalement un seul point solution parmi un ensemble possible. Les solveurs d'équations itératifs ont alors été étendus à un problème plus général qui est l'optimisation. Le dimensionnement d'un circuit analogique s'exprime finalement comme un problème d'optimisation. Nous allons en présenter les concepts.

4.3 Formulation générale d'un problème d'optimisation

L'optimisation entre en jeu dans beaucoup de domaines scientifiques et la recherche de performance, de rentabilité et de fiabilité est au cœur des processus de conception et de production des systèmes. Avant de définir ce qu'est un problème d'optimisation multi-objectifs, nous allons présenter le problème d'optimisation sous sa forme mathématique ainsi que les notions de minima local et global qui sous-tendent à ce problème.

4.3.1 Formulation mathématique d'un problème d'optimisation

Partant de l'existence d'une relation $p = f(a)$, on souhaite contraindre p à satisfaire un objectif tel que : $p \in [p_{min}, p_{max}]$. Cet objectif impose de trouver l'ensemble des variables d'ajustement A_S tel que $\forall a \in A_S$:

$p = f(a) \in [S]$ avec $[S] = [p_{min}, p_{max}]$.

Trouver une variable a qui satisfasse une telle condition repose aujourd'hui sur un processus de recherche itératif. Ce processus consiste à faire tendre un point de performances vers un point des spécifications tel que l'image par la fonction f soit minimale.

Le problème précédent peut être reformalisé par :

$$\begin{cases} \text{Minimiser} \quad f(a) \quad a \in R^n \\ \text{tel que :} \ f(a) = (S - f(a))^2 \end{cases}$$

C'est le choix de la fonction f qui par sa caractéristique permet via un problème d'optimisation de résoudre un problème de résolution numérique d'équations. La fonction f est aussi appelée la fonction de coût du problème d'optimisation.

Remarque : Nous ne parlerons que de minimisation car la maximisation d'une fonction $f(a)$ est équivalent à la minimisation de la fonction $[-f(a)]$.

Les problèmes pratiques imposent souvent des hypothèses préalables pour la modélisation mathématiques. Résoudre un système d'équations en respectant ces hypothèses impose de contraindre le solveur à ne rechercher des solutions que dans le sous-espace des variables où les hypothèses sont satisfaites ! La généralisation de ce cas conduit aux problèmes d'optimisation avec contraintes que l'on formalise par :

$$equ.(1)\begin{cases}\text{Minimiser} \quad f(a) \quad a \in R^n \\ \text{sous les contraintes :} \\ g_i(a) \geq 0 \quad i = 1,...,m \\ h_i(a) = 0 \quad i = 1,...,p\end{cases}$$

On distingue deux classes de contraintes. Les contraintes d'inégalités que nous exprimerons sous la forme de fonctions $g_i(a)$ avec $i=1, \ldots, m$ et les contraintes d'égalités désignées par les fonctions $h_i(a)$ avec $i=1, \ldots, p$.

Résoudre un problème d'optimisation sous contraintes suppose d'une part, de satisfaire l'ensemble des contraintes exprimées par les fonctions g_i et h_i, et d'autre part de trouver un point qui est le minimum de la fonction de coût f. Satisfaire toutes les contraintes du problème (CSP) permet de rejeter immédiatement les points solutions infaisables. Par cette approche, nous obtenons le sous-espace faisable associé aux contraintes. Cela revient à limiter l'exploration à une zone d'intérêt plus réduite afin de proposer des conditions de départ proche de l'optimum. Il est par ailleurs fortement recommandé d'augmenter le nombre de contraintes associée à un problème d'optimisation afin « d'isoler » au mieux le sous-espace faisable [Cha-92, Kun-00, Cha-01, Ber-05]. Aborder un problème d'optimisation par la résolution préalable du CSP permet finalement de faire gagner un temps de recherche dans les processus itératifs de minimisation [Mue-05].

Classification des problèmes d'optimisation

Si le problème d'optimisation consiste à minimiser une seule fonction de coût, on parlera alors de problème mono-objectif. Dans le cas contraire, il existe au moins deux objectifs à satisfaire, le problème d'optimisation devient un problème multi-objectifs (ou multi-critères).

Par exemple, si la conception d'un circuit intégré est uniquement conditionnée par la minimisation de sa consommation énergétique, le problème d'optimisation est dit mono-objectif et se contente alors de minimiser une seule fonction de coût. Cette fonction de coût fait alors intervenir les variables d'ajustement les plus influentes au regard du critère de consommation. Si par contre, la conception du circuit intégré est également conditionnée par la maximisation de la puissance de calculs, le problème d'optimisation devient alors multi-objectifs.

Traditionnellement, les problèmes d'optimisation sont classés en plusieurs catégories selon les propriétés de la fonction de coût et des fonctions de contraintes :

Fonction de coût (mono-objectif)	**Fonctions de contraintes**	**Terminologie**
Fonction linéaire	Pas de fonctions	Optimisation linéaire sans contraintes
Fonction non linéaire		Optimisation non linéaire sans contraintes
Fonctions linéaires		Optimisation linéaire avec contraintes
Fonctions non linéaires		Optimisation non linéaire avec contraintes

4.3.2 Optimisation locale ou globale

Une fonction de coût présente généralement un grand nombre de minima. Un exemple d'une telle fonction à une dimension est présenté dans la figure 4-2. Il s'agit d'une fonction à une variable réelle *a* dont on cherche à extraire le minimum.

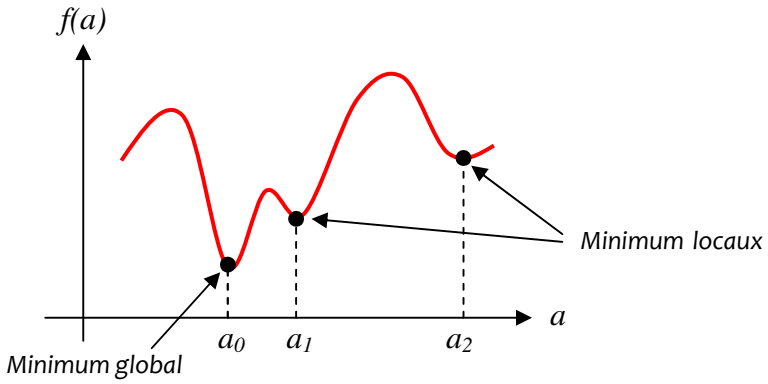


Figure 4-2 Représentation d'une fonction de coût à une variable présentant des minima locaux et un minimum global.

La fonction présente plusieurs minima que nous distinguons par la terminologie de minimum local ou minium global ; nous les définissons de la façon suivante [Bie-06] :

- $f(a^*)$ est un minimum local si :
$$\exists \varepsilon > 0 / \forall a \in A : \quad \| a - a^* \| < \varepsilon \Rightarrow f(a) \geq f(a^*)$$
- $f(a^*)$ est un minimum global si :
$$\forall a \in A : \quad f(a) \geq f(a^*)$$

L'objectif final du processus d'optimisation consiste à trouver le minimum global de *f*, et d'éviter d'être piégé dans un minimum local.

Nous pouvons partager les méthodes d'optimisation en deux catégories : celles qui trouvent un minimum local le plus proche du point de départ (ces méthodes utilisent des méthodes d'optimisation locales), et celles qui s'efforcent de déterminer un minimum global (ces méthodes utilisent des méthodes d'optimisation globale).

Il existe un grand nombre de méthodes issues des deux catégories précédentes et les énumérer toutes nous éloignerait de notre sujet. Nous nous contenterons de citer les méthodes les plus populaires.

Parmi les méthodes locales on trouve typiquement la descente de gradient. Ce type de méthodes est « sensible » aux conditions de départ. En effet, elles convergent systématiquement vers le minimum situé dans le bassin d'attraction dans lequel est placé le point de départ.
Parmi les méthodes globales, on pourra citer :

- la méthode du recuit simulé [Gie-90] ;
- la méthode posynomiale [Dae-03] ;
- la programmation géométrique [Ste-07] ;
- ou encore les méthodes évolutionnaires qui font intervenir les algorithmes génétiques.

L'efficacité des méthodes locales est fortement liée aux conditions initiales et limite en cela leur champ d'application. En revanche, elles sont nettement plus rapides que les méthodes globales. Cela démontre l'intérêt à résoudre un problème par une double optimisation, c'est-à-dire globale puis locale, en tirant le meilleur parti de chacune d'elles.

Dans un problème d'optimisation globale, une notion qui va jouer un rôle très important est celle de convexité. Nous allons voir dans l'exemple suivant que la présence de minima locaux est liée directement à la notion de non convexité d'un espace.

Considérons le sous-espace *A* de la figure 4-3 délimité par la fonction $f(a_1,a_2) = a_1^2 + a_2^2$. Par définition, le sous-espace *A* est dit convexe si pour tous points (a_1, a_2) de ce sous-espace et pour tous $\alpha \in [0,1]$, on observe :

$$f(\alpha a_1 + (1-\alpha)a_2) \leq \alpha f(a_1) + (1-\alpha) f(a_2)$$

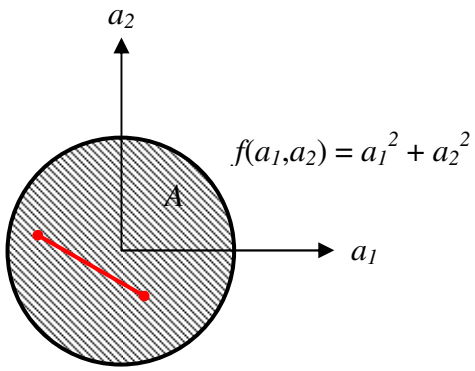


Figure 4-3 Délimitation par une fonction *f* d'un sous-espace convexe

En d'autres termes, le sous-espace *A* est dit convexe si pour tous points a_1 et a_2 de ce sous-espace, le segment liant ces points est contenu dans le sous-espace, ce qui est le cas sur la figure 4-3.

Le sous-espace est par conséquent convexe et traduit la présence d'un seul minimum. En effet, la fonction $f(a_1,a_2) = a_1^2 + a_2^2$ possède un extremum unique en $(a_1,a_2) = (0,0)$ qui est un minimum local. Comme ce minimum local est unique, il sera alors aussi le minimum global. Cet exemple illustre qu'une méthode locale peut donc être employée à la recherche d'un minimum global lorsque le problème est convexe. Cette propriété est par ailleurs largement exploitée par les méthodes posynomiales qui cherchent à transformer un espace non convexe en un espace convexe. Ce point souligne l'importance qu'ont encore les méthodes locales.

La convexité est une propriété relative à une certaine formulation d'un problème. Ainsi beaucoup de problèmes que l'on peut supposer non-convexes admettent en fait une formulation convexe [Hen-07]. La convergence vers un minimum global ne pourra être démontrée qu'avec des hypothèses de convexité. Un certain nombre de méthodes d'optimisation s'inspire de cette propriété pour faire converger leur processus de recherche vers un minimum global (méthode posynomiale, programmation géométrique).

L'exemple de la figure 4-4 illustre le cas d'un sous-espace non convexe. Considérons la fonction suivante :

$$f(a_1,a_2) = (a_1^2 - 4).(a_1^2 - 1).a_1 + (a_1 - 0.5)^2 + 4.(a_2 - 0.5)^2$$

Cette fonction est représentée en différents plans (courbes de niveaux à valeurs de *f* constantes) et démontre qu'il est possible de relier au moins deux points par un segment dont une partie est en dehors du domaine.

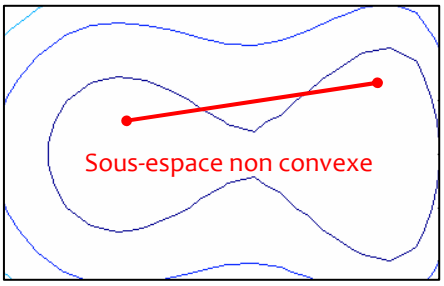


Figure 4-4 Délimitation par la fonction *f* d'un sous-espace non convexe

Une représentation plus détaillée par courbes de niveaux de l'espace permet de mettre en évidence sur la figure 4-5 la présence de plusieurs minima et fait apparaître par conséquent le risque de piège dans un minimum local.

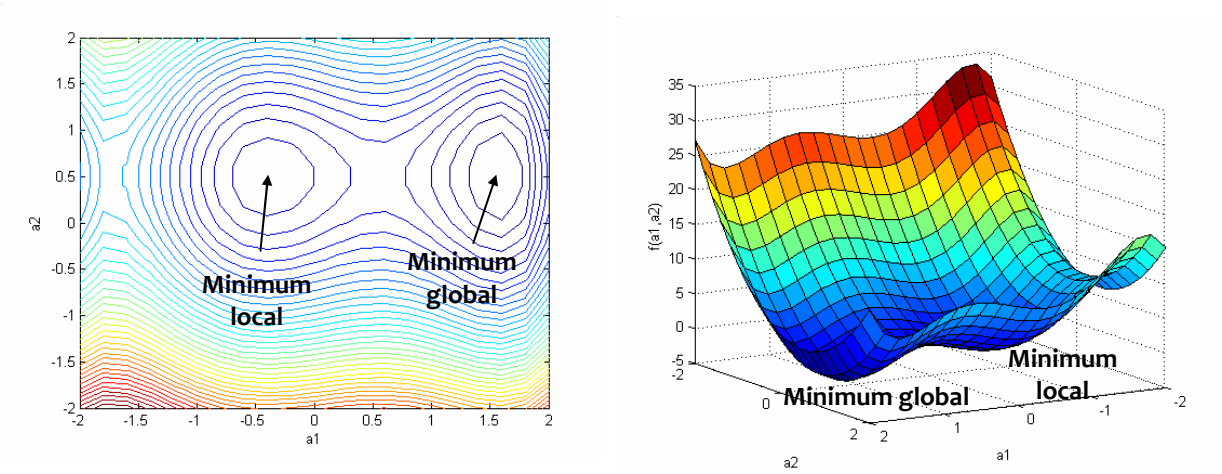


Figure 4-5 Présence de deux minima sur la fonction *f*

4.3.3 Optimisation globale multi-objectifs

Les problèmes d'optimisation issus de problématiques concrètes sont la plupart du temps de nature multi-objcetifs.

Le problème d'optimisation fait alors intervenir un vecteur de fonctions objectifs et consiste à :

$$\begin{cases} \text{Minimiser} \quad F(a) = (f_1(a), f_2(a), ..., f_k(a)) \quad a \in R^n \\ \text{sous les contraintes :} \\ g_i(a) \geq 0 \quad i = 1, ..., m \\ h_i(a) = 0 \quad i = 1, ..., p \end{cases}$$

Il existe plusieurs méthodes pour traiter ce type de problème. La méthode séquentielle par exemple, transforme un problème multi-objectifs en un problème mono-objectif en considérant chaque fonction de coût séparément et les unes après les autres [Coe-96]. Une liste de priorité

des fonctions de coût est alors préalablement établie et l'opération consiste à optimiser un objectif sans dégrader les valeurs déjà obtenues pour les objectifs plus prioritaires. Cependant, il n'est pas toujours possible de trouver un ordre d'importance entre les objectifs. Il est alors nécessaire de rechercher les solutions de meilleur compromis entre ces objectifs.
Ces critères peuvent être en conflits avec d'autres et l'exemple de la conception du circuit sous les critères de minimisation de la consommation et maximisation de la puissance de calculs en est un exemple.

Considérons la situation illustrée sur la figure 4-6 où deux fonctions $f_1(\boldsymbol{a})$ et $f_2(\boldsymbol{a})$ présentent un choix de plusieurs optima. Pour retenir un seul optimum, des critères additionnels doivent être posés. Si l'on relève les points optima selon des critères de minimisation, alors les points (a_{11}, a_{21}) et (a_{12}, a_{22}) répondent tous deux à ces objectifs mais par rapport à une seule variable. Le point (a_{11}, a_{21}) correspond au critère $min(f_1(\boldsymbol{a}))$ selon la dimension de a_2 alors que le point (a_{12}, a_{22}) correspond au critère $min(f_2(\boldsymbol{a}))$ selon a_1.

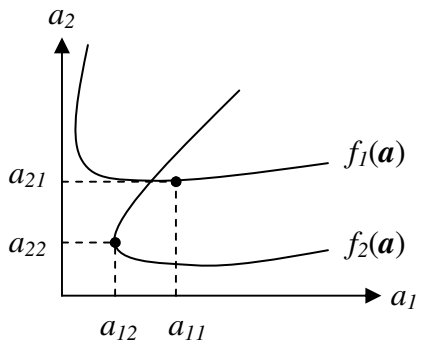


Figure 4-6 Illustration d'une situation de compromis entre deux optima

Une réponse à cette problématique est alors de faire apparaître l'idée d'un compromis entre objectifs tel que le propose la méthode à objectifs pondérés. Elle consiste à attribuer un coefficient à chacune des composantes $f_i(a)$ de la fonction de coût et à minimiser leur somme :

$$\min \sum_{i=1}^{k} w_i f_i(a) \quad \text{où} : w_i \geq 0$$

La méthode de pondération est relativement simple à utiliser mais il est assez difficile de fixer a priori les valeurs des poids associés à chaque objectif. Il est par ailleurs impératif de normaliser les objectifs lorsque ceux-ci n'ont pas les mêmes dimensions physiques. De plus, cette méthode n'est pas adaptée aux problèmes non convexes qui empêchent l'algorithme de converger vers le minimum global [Rei-03]. Le champ applicatif de ces méthodes se trouve alors réduit mais illustre, grâce au principe de pondération, l'idée de compromis possibles. A ce stade, rien ne justifie qu'une solution soit forcément « meilleure » ou « préférable » qu'une autre. Cet ensemble de solutions non dominées est connu comme l'ensemble des solutions Pareto Optimales.

La figure 4-7 illustre par exemple le cas d'un problème d'optimisation bi-objectifs donnés par la minimisation des fonctions $f_1(\boldsymbol{a})$ et $f_2(\boldsymbol{a})$. Un échantillon de réponses à ces deux fonctions est représenté dans l'espace objectif. Parmi les réponses du problème, un certain nombre est dominé par d'autres dans le sens où ces dernières offrent un meilleur compromis par rapport aux objectifs visés ($min[f_1(\boldsymbol{a})]$ et $min[f_2(\boldsymbol{a})]$). Ces réponses sont alors désignées par le terme de points réalisables (cercles). Dans le cas contraire, les réponses sont aussi les solutions au problème de minimisation et sont désignées par l'ensemble des solutions Pareto optimale (disques). Toute solution de l'ensemble Pareto peut être considérée comme optimale puisque aucune

amélioration ne peut être faite sur un objectif sans dégrader la valeur relative à un autre objectif. Ces solutions forment le front de Pareto. La méthode de pondération permet de construire le front de Pareto en exécutant plusieurs optimisations successives avec des facteurs de pondération différents. Ainsi, la droite d_o propose avec la pondération (w_1,w_2) le point a_o comme solution du meilleur compromis, tandis que la droite d_1 propose avec la pondération (w_1',w_2') le point a_1.

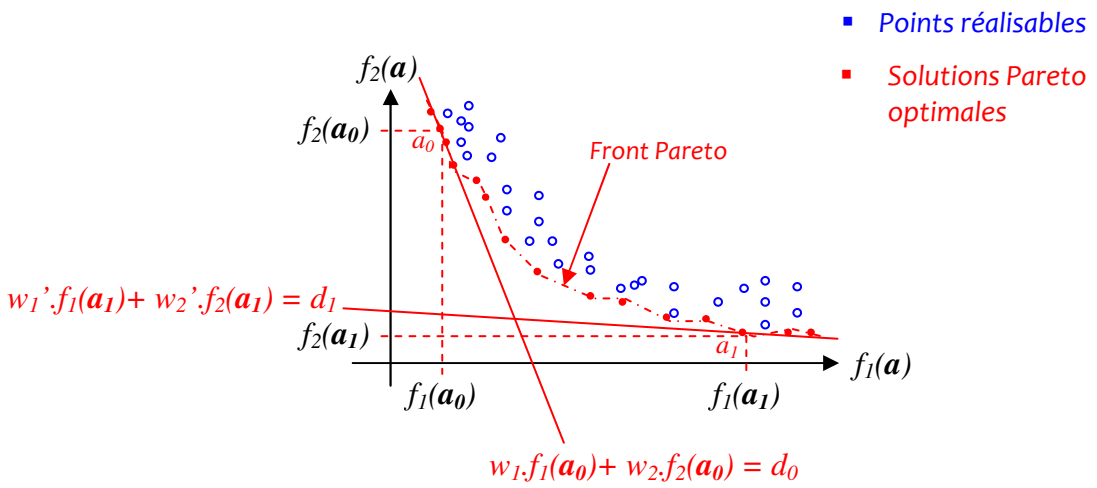


Figure 4-7 Illustration des solutions par front de Pareto

Ainsi, en fin de traitement, l'algorithme fournit un ensemble de solutions représentant toutes un compromis optimal par rapport aux objectifs de minimisation du problème. Le choix de la solution finale revient donc à l'utilisateur, qui doit choisir parmi l'ensemble fourni la solution qui lui convient le mieux.

Nous venons de poser les concepts en optimisation. Ces concepts sont mis en œuvre dans les problèmes de dimensionnement de circuits selon différentes approches. Nous allons maintenant les présenter.

4.4 L'optimisation dans le processus de dimensionnement d'une architecture

Avec la complexité croissante des circuits analogiques, le problème de leur dimensionnement nécessite l'apport d'outils d'aide à la conception pour répondre à la réduction du temps de conception. En effet, cette complexité, qui se traduit notamment par l'augmentation du nombre de variables de conception, limite les capacités de l'ingénieur à prendre la bonne décision dans des délais « courts ». En supposant que pour chaque variable de conception, il existe un certain nombre de solutions potentielles à visiter, alors multiplié par le nombre de variables, le temps de conception total devient excessif pour un processus manuel. L'approche basée sur la connaissance qui repose sur la culture du concepteur fait donc peu à peu place à l'approche par optimisation. L'approche par optimisation permet d'automatiser la procédure de recherche en calculant les valeurs des variables de conception à l'aide de méthodes numériques qui fournissent une solution optimale ou proche de l'optimal.
Calculer les valeurs de ces variables nécessite d'évaluer la performance d'un circuit pour en mesurer l'écart par rapport aux spécifications. L'outil de dimensionnement se compose typiquement d'un moteur d'optimisation et d'évaluation de performance [Rut-07]. La figure 4-8 illustre les différentes étapes nécessaires à la recherche d'une solution optimale en phase de dimensionnement.

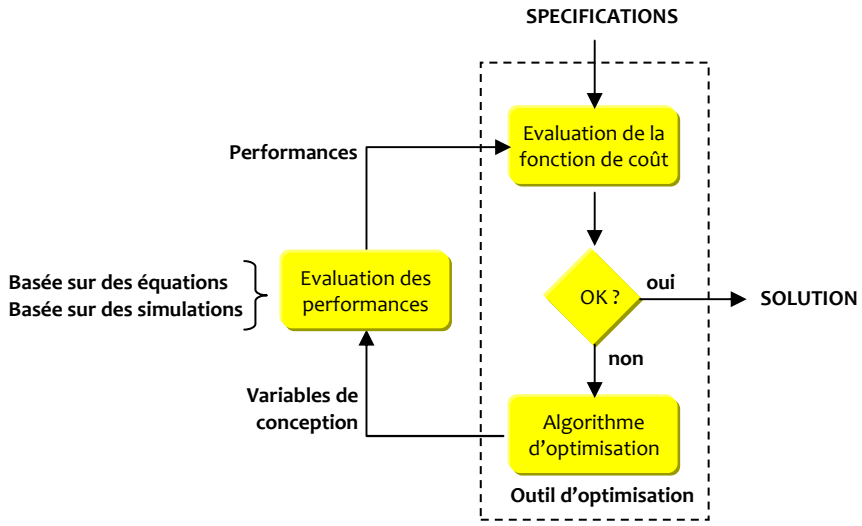


Figure 4-8 Les étapes de recherche de la solution optimale en phase de dimensionnement

On distingue alors deux approches d'optimisation dont chacune se différencie par la façon de décrire le circuit pour en mesurer les performances. Dans le cas où le circuit est directement décrit par un jeu d'équations, l'optimisation porte alors le nom « d'optimisation basée sur des équations ». Si le circuit est décrit à partir d'une interconnexion de modèles de composants, on parlera alors « d'optimisation basée sur la simulation ». Les performances sont alors calculées à partir de l'une ou l'autre de ces descriptions. Le coût de ces performances est ensuite comparé aux spécifications qui sont formulées par le concepteur. Tant que le coût reste supérieur à un seuil d'acceptation, de nouvelles valeurs sont attribuées aux variables de conception à partir de l'algorithme d'optimisation. Ces valeurs sont calculées en fonction de critères de minimisation du coût des performances. Le problème de dimensionnement se base finalement sur un processus itératif qui consiste à trouver les valeurs des variables de conception telles que les performances correspondent aux spécifications à un seuil de tolérance près.

Les deux approches d'optimisation par simulation et par équations sont maintenant présentées.

4.4.1 L'approche par simulation

En optimisation basée sur la simulation, les performances du circuit sont calculées à partir d'un simulateur. Les outils basés sur cette approche peuvent manipuler une grande variété de circuits analogiques et tout circuit simulable devient en principe optimisable. Cette approche est donc universelle. La description des modèles de composants utilisés permet de reproduire des caractéristiques électriques calculées en bon accord avec les caractéristiques électriques réelles. Ces modèles reproduisent « fidèlement » les phénomènes physiques (régimes de fonctionnement des transistors, champs électriques internes, …) mais nécessitent un très grand nombre de paramètres (au-delà de 300) que les fabricants de circuits doivent garantir. L'approche par simulation permet donc d'établir un lien très fort avec la technologie des composants employés. La précision dans les résultats est la principale qualité recherchée par l'utilisateur de ce type d'approche.
L'outil de simulation le plus employé par les concepteurs de circuits analogiques est SPICE. Il permet de résoudre des équations algébriques différentielles qui caractérisent le circuit en utilisant les techniques traditionnelles d'analyse numérique. Le principal défaut est lié au temps de résolution qui augmente avec la taille du circuit analysé. L'approche par simulation présente un temps de simulation multiplié par le nombre d'itérations nécessaire à la convergence du

processus d'optimisation. Un autre point délicat est que cette approche nécessite d'avoir sélectionné une topologie sans savoir si une solution au dimensionnement existe ! Ce risque peut donc conduire à un accroissement du temps d'analyse. Enfin, la description d'un circuit par ce type d'approche ne permet pas d'avoir un regard sur la formulation mathématique des performances du circuit. L'approche par simulation correspond à une approche type « boîte noire » peu adaptée à une analyse au niveau système. Ceci fait donc de SPICE une application limitée à ne pouvoir simuler que des circuits décrits structurellement.

Des outils d'aide au dimensionnement selon cette approche ont été développés. L'outil FRIDGE [Med-94] par exemple utilise la méthode du recuit simulé (*simulated annealing*) dans sa phase de recherche globale. L'outil MAELSTROM [Kra-99] effectue une optimisation globale à partir d'un algorithme combiné génétique/recuit simulé dans le cadre de la sélection de topologies. Cet outil peut être associé sur la même plateforme à l'outil ANACONDA [Phe-00] qui procède à une recherche par une méthode évolutionnaire des variables d'ajustement d'un circuit. Différents circuits candidats sont donc de cette façon évalués en vue d'une sélection de topologies. Enfin, l'outil MOJITO [Mcc-07] met en ouvre une optimisation globale multi-objectifs à partir d'un algorithme évolutionnaire permettant de d'analyser simultanément plus d'une centaine de topologies différentes.

4.4.2 L'approche par équation

L'optimisation du dimensionnement basée sur des équations calcule les performances d'un circuit à partir d'un jeu d'équations analytiques. Ce jeu d'équations définit alors le modèle de performances du circuit. L'exemple du filtre passe-bas (*cf.* §4.2) est caractérisé par un modèle de performances exprimant son gain et sa phase. Le jeu d'équations d'un modèle de performances peut être généré manuellement ou à partir d'outils d'aide comme un analyseur symbolique. Ce point sera abordé plus loin dans le paragraphe 5 de ce chapitre.

La précision du calcul des performances dépend de la finesse de description des modèles de performances. En pratique et dans une première approche, les équations de ces modèles sont simplifiées pour bénéficier d'une vitesse de traitement plus élevée. Le circuit décrit par un modèle de performances simplifié est alors perçu comme plus générique car moins lié aux limitations techniques de la réalisation. Le bénéfice d'un temps de calcul plus court est alors contre balancé par une baisse de la précision des évaluations des performances. L'approche par équations se détache donc davantage de la technologie employée et la rend indépendante de celle-ci. On progresse ainsi plus en douceur dans la façon d'introduire les niveaux de complexité du problème à optimiser.

4.4.3 Comparaison des deux approches

Le tableau ci-dessous permet de synthétiser les avantages et inconvénients des deux approches présentées auparavant [Agg-07]. La métrique de comparaison est déclinée selon cinq critères : la précision, l'effort de préparation du traitement, le temps incluant la préparation pour l'optimisation et le calcul, l'aptitude pour une conception système et l'aptitude à traiter la robustesse.

Critères	Optimisation par simulations	Optimisation par équations
Précision	+ + (très bonne précision) -- (convergence vers optimum global incertain)	-- (peu précis) ++ (permet de trouver un optimum global)
Effort de préparation	+ + (Effort faible)	-- (Effort important dans l'écriture du jeu d'équations)
Temps de préparation	+ + (temps de préparation faible) - - (convergence globale lente)	-- (temps de conditionnement à l'optimiseur important) ++ (temps de calculs faible)
Aptitude système	-- (nécessite des itérations entre deux solveurs niveau système et cellule)	+ + (équations extensibles à la description système)
Aptitude robustesse	-- (évaluation statistique longue) + + (évaluation statistique juste)	- - (relations entre paramètres technologiques et performances très difficiles)

Le tableau fait clairement apparaître la disparité entre la précision du modèle et le temps de calculs. Le tableau précise le manque d'aptitude des optimiseurs à base de simulations dans l'évaluation des performances au niveau système. Ce constat est repris dans la partie qui suit où un inventaire des besoins en matière de conception de circuits mixtes permet de positionner spécifiquement l'approche par équations.

4.4.4 Exploitation de l'approche par équation

Aujourd'hui, la conception de circuits intégrés mixtes doit répondre principalement à deux besoins [Gie-02] :

- Le besoin d'un environnement d'exploration architecturale à signaux mixtes permettant de faciliter l'analyse et la comparaison de plusieurs solutions architecturales en terme de performances, de surface et de consommation.
- Le besoin de disposer d'un environnement de modélisation à haut niveau d'abstraction et d'estimation du rapport surface/énergie pour pouvoir comparer différentes solutions au niveau système.

Ces besoins imposent une approche de la conception des circuits mixtes qui peut être décomposée en trois niveaux hiérarchiques si l'on prend en compte l'expression du concept jusqu'au dimensionnement d'une cellule (figure 4-9). En partant de l'idée qui conceptualise le système à concevoir, un premier niveau « système » décrit l'architecture fonctionnelle globale. Un second niveau décompose ensuite individuellement chaque bloc en sous-blocs : ce stade correspond à la conception architecturale. Pour chacun de ces sous-blocs une topologie est alors sélectionnée. Cette sélection de topologie est réalisée au troisième niveau et est accompagnée d'une phase de dimensionnement : c'est la conception au niveau cellule. Chaque niveau s'appuie sur un modèle de performances avec ses paramètres. Un test d'adéquation performances - spécifications est alors effectué pour répondre aux objectifs visés. Si le test est validé, le niveau hiérarchiquement inférieur est ensuite exécuté et le test réitéré. Il est nécessaire de souligner que lorsqu'un choix d'ordre architectural ou structurel (sélection d'une topologie) est effectué, une remontée des performances est réalisée sur l'ensemble des niveaux supérieurs. Cette remontée reprend les tests de compatibilité avec les spécifications jusqu'au niveau sous-système.

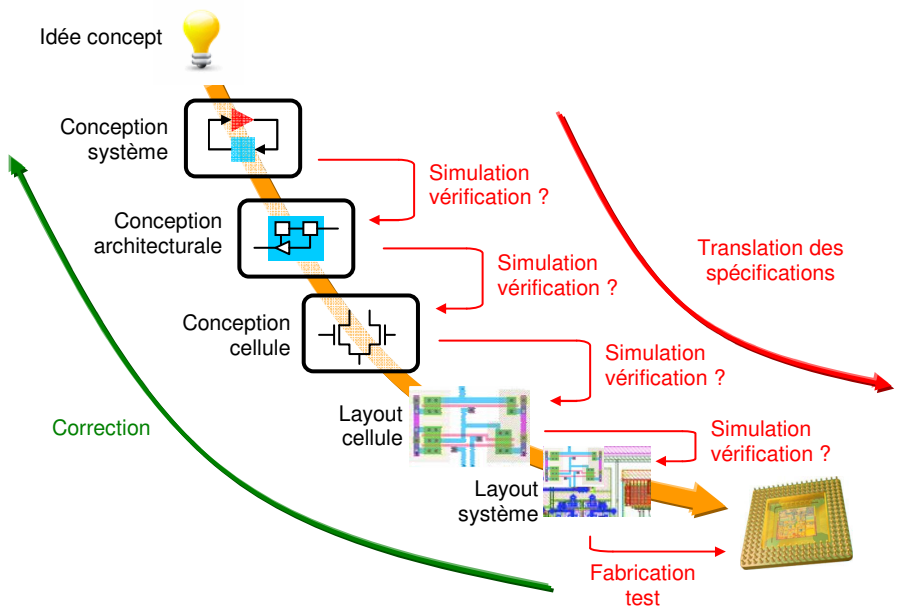


Figure 4-9 Décomposition du flot de conception « Top-Down » d'un circuit à signaux mixtes

Une telle approche nécessite par conséquent que des modèles comportementaux au niveau système et architectural soient développés. Cela suppose donc de pouvoir disposer d'un jeu d'équations qui décrivent les performances qui puissent être ensuite comparées aux spécifications. Dans ce cas, l'approche par équations est alors plus rapide comparativement à l'approche par simulations6. Le fait de disposer de modèles non liés à la technologie offre la possibilité d'explorer rapidement les performances de plusieurs architectures. Ceci permettant d'éliminer au plus tôt les topologies inconsistantes et qui induiraient de multiples itérations dans le processus d'optimisation.
L'approche par équations permet de réduire le temps de recherche de solution et d'accélérer l'exploration architecturale. Il s'agit là d'un objectif global assigné aux outils d'aide à la conception. Ce point pour nous est fondamental dans notre approche pour résoudre le problème de l'efficacité de la simulation par le compromis précision – simplicité des modèles.
Cette approche a cependant un prix qui est celui du premier investissement en terme de temps pour le développement des modèles. Il est en effet nécessaire de pouvoir disposer de modèles de tout niveau partant du plus bas au plus haut niveau d'abstraction. Une réponse peut être proposée à cette critique qui est celle de la génération automatique d'équations par calcul symbolique.

4.5 Les outils en génération d'équations

Les techniques de génération automatique de fonctions, connues sous l'appellation d'analyses symboliques, se sont considérablement développées ces dernières années et contribuent à une automatisation du processus de conception d'un circuit à signaux mixtes.
Ces techniques permettent à partir d'une description d'un circuit, d'en générer des expressions qui peuvent être utilisées avantageusement dans de nombreuses situations. Par exemple, cette technique présente l'intérêt de fournir des expressions que le concepteur pourra interpréter pour étudier le comportement d'un circuit. Il pourra ainsi juger de l'importance relative des différents

6 Le temps de simulation étant proportionnel au nombre de composants, en remontant vers une simulation globale au niveau système, le nombre de composants augmente entraînant avec lui un accroissement du temps de simulation

paramètres et procéder à d'éventuelles simplifications des expressions. Une autre application consiste à se servir des programmes d'analyse symbolique pour générer automatiquement des modèles analytiques dans le cadre de la synthèse analogique de circuits. Par exemple, il peut être intéressant à partir de topologies de circuits nouvellement créées[7] d'en développer les modèles de performances.

4.5.1 Principe de l'analyse symbolique

Le processus de génération d'expressions littérales est schématisé sur la figure 4-10 [Gie-94] et se décompose en deux étapes principalement. Lors de la première étape, le circuit analysé est décrit par sa netlist selon une syntaxe de type SPICE par exemple. Dans l'hypothèse d'un circuit analogique non-linéaire, le modèle du circuit équivalent petits signaux linéarisé est ensuite généré. Une analyse est alors effectuée selon deux méthodes possibles : la méthode algébrique ou la méthode géométrique [Wam-98]. Ces deux méthodes fournissent l'équation brute mais elle reste à ce stade encore peu adaptée à un traitement par un outil d'optimisation. Lors de la seconde étape, une simplification des équations symboliques est effectuée. Cette simplification est motivée par l'observation que la plupart des caractéristiques d'un circuit, telles que le gain ou la marge de phase, sont influencées par un nombre limité de variables [Fer-97]. Cette opération de simplification permet alors de réduire la complexité des expressions au bénéfice d'une réduction du temps de leur évaluation numérique lors du processus d'optimisation. L'intervention d'un expert est d'ailleurs indispensable à ce stade du processus de simplification. Enfin, l'expression simplifiée est ensuite compilée et rendue exploitable par le bloc d'évaluation numérique.

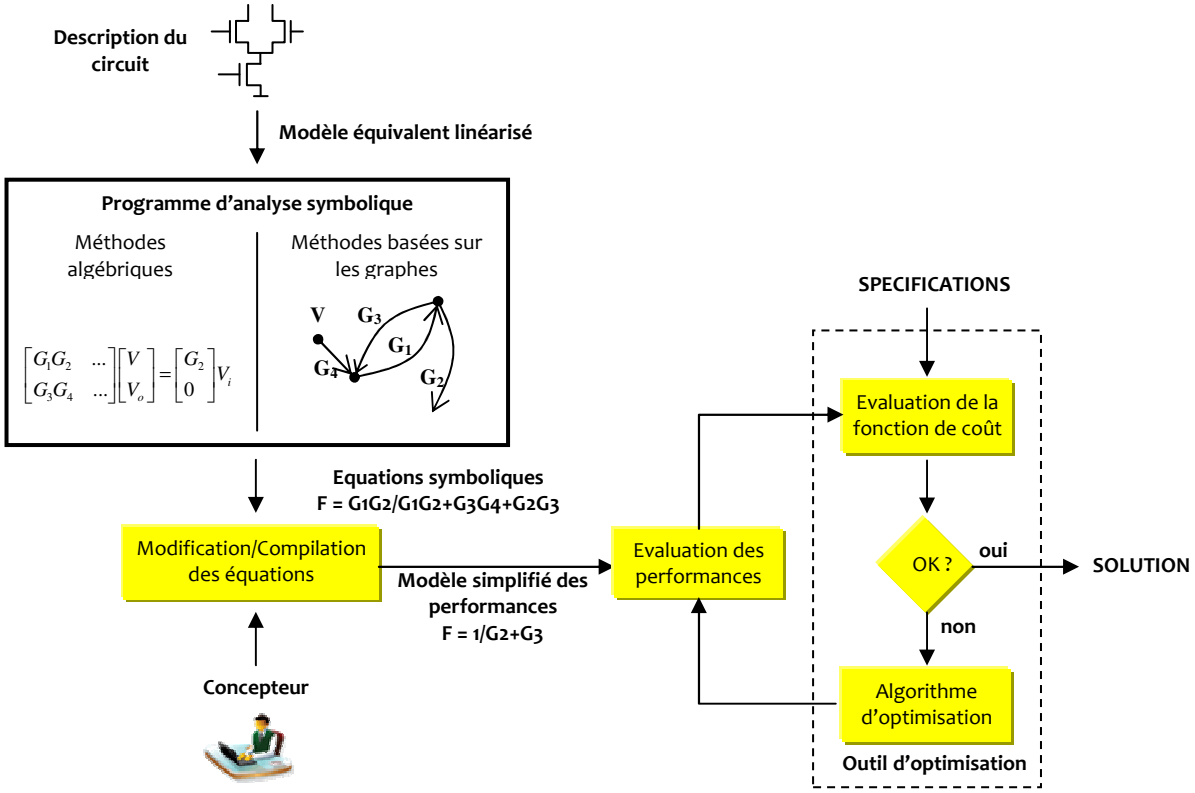


Figure 4-10 Procédé de génération d'équations symboliques pour l'optimisation en synthèse analogique

7 Des outils existent selon un mécanisme de Branch and Prune [Gie-89]

4.5.2 Les outils en analyse symbolique

Depuis les années 1980 et le développement des ordinateurs, les techniques de génération d'équations dans le domaine des circuits électroniques ont été implémentées dans des machines. Ceci s'est illustré avec notamment le succès de l'analyseur symbolique ISAAC [Gie-89] ou ASAP [Fer-90]. L'outil DONALD permet de transformer des expressions en sélectionnant des variables indépendantes et en proposant un ordre d'exécution des expressions en phase d'évaluation des performances. Des outils implémentent à la fois les programmes d'analyse symbolique et le processus d'optimisation. L'algorithme OPTIMAN [Gie-90] par exemple utilise la méthode d'optimisation du recuit simulé pour déterminer les extrema de la fonction de coût. Le système ARIADNE permet alors de regrouper dans un environnement unique les trois outils ISAAC, DONALD et OPTIMAN [Swi-93, Deg-90]. L'exploration de circuit est également une possibilité permettant, à partir d'une topologie donnée, de mesurer l'influence d'une modification par l'ajout d'un composant. Le programme PROLOG par exemple génère selon cette méthode des expressions analytiques à partir de topologies qui ont été modifiées.

La génération automatique de modèles contribue à la réduction du temps de conception d'un circuit mixte. Elle contribue également à fournir une bibliothèque de cellules avec des modèles qui pourront être réutilisés.

4.5.3 La réutilisation des modèles

La réutilisation de modèles peut aider le concepteur dans sa façon de gérer la complexité d'un système et lui permettre de tenir des délais liés à l'exécution du processus de conception. On définit par le terme de réutilisabilité (*Reuse*) la démarche consistant à reprendre l'expérience aboutie d'une conception (de composant, de circuit ou de fonction) pour l'appliquer à un autre projet de conception. On désigne alors par le terme de composant virtuel (Virtual component) ou de propriété intellectuelle (*Intectual property*) ces entités conçues dans le but d'appliquer ce concept de réutilisabilité. Le degré de réutilisabilité se mesure alors à la part de travail du projet initial pouvant être transférée au projet cible.

Deux scenarii justifient alors l'emploi de composants réutilisables :

- La reformulation des spécifications de conception lorsque de nouveaux objectifs doivent être atteints.
- La migration ; le procédé de migration intervient lorsqu'un circuit doit être implémenté selon un processus de fabrication différent de celui pour lequel il a été conçu originellement.

L'approche du reuse en analogique comparativement à celle du numérique est significativement bien plus difficile à mettre en œuvre. Les outils sont en effet pour le moment moins nombreux en analogique et reposent sur des méthodologies insuffisamment structurées. Un appoint est alors indispensable et nécessite toute l'expérience et le savoir faire du concepteur. Un autre point d'achoppement concerne la difficulté et l'effort nécessaire à formaliser l'ensemble des connaissances (spécifications, modèles, expérience de la phase de développement du layout, …). Ce point est particulièrement critique sachant l'accroissement de la vitesse de changement de la technologie des circuits intégrés et de l'hétérogénéité des applications à concevoir. Ceci souligne l'importance de disposer d'outils automatisés qui soient généralistes et universels.

Un certain nombre d'approches académiques et non commercialisées ont été depuis proposées :

L'approche introduite par l'outil VSIA [VSIA-99] propose une structure permettant de mieux définir l'interface entre les développeurs d'IP et leurs utilisateurs. On retrouve l'outil ANACONDA [Phe-00a] ainsi que MAELSTROM [Kra-99].
Un certain nombre de méthodologies de conception incluent conjointement les effets introduits lors de la phase de layout, toutes basées sur une synthèse par connaissance. On y trouve COMDIAC [Por-97] et CAIRO [Des-99, Ngu-06] qui tiennent compte lors de la phase de dimensionnement de la technologie employée et des informations physiques du layout sous la forme de procédures codées. .

4.6 Notre approche dans le dimensionnement d'une architecture

Nous présentons les orientations de notre approche dans le dimensionnement d'architectures en relevant un certain nombre de constats tirés des techniques actuelles.

4.6.1 Bilan et observations

Le dimensionnement d'une architecture est une tâche longue, et qui réclame des ressources de calculs importantes pour au final ne retenir qu'un seul point dans l'espace des variables d'ajustement. Ce jeu de variables d'ajustement est le plus souvent obtenu à partir d'un processus d'optimisation global sous contraintes. Dans l'hypothèse de l'existence d'une solution, extraire la solution optimale d'un problème d'optimisation nécessite typiquement plusieurs itérations. La durée et la difficulté de l'optimisation augmentent rapidement avec le nombre de variables d'ajustement à optimiser [Gie-05a, Rut-07] c'est-à-dire avec le nombre de degré de libertés. Il convient donc d'examiner la possibilité de limiter l'étendue ou la dynamique de certaines de ces variables d'ajustement.
Les techniques ponctuelles présentent une autre limite, celle de retourner un point solution avec peu d'information sur son voisinage. Ce manque d'information rend notamment difficile une analyse de robustesse face aux dérives de fabrications qui pénalisent alors le rendement de production. Aujourd'hui, l'impact de ces dérives est typiquement étudié avec des simulations du type Monte-Carlo qui apporte une réponse à ce manque de visibilité mais au détriment d'un temps d'analyse plus long. De plus, les résultats ne peuvent pas être capitalisés en cas de recherche d'un nouvel objectif et l'analyse doit être reprise à son point de départ.

Construire un espace de conception plutôt que d'extraire une solution unique semble être une approche plus ouverte. L'approche par front de Pareto apporte une réponse limitée puisque seule une surface « optimale » est extraite. L'idée d'exprimer un volume solution et d'utiliser ce volume pour ensuite en déduire une information sur la robustesse aux variations technologiques semble plus adaptée.

4.6.2 Notre approche

Notre objectif est de développer une approche exploratoire permettant de limiter le temps d'analyse nécessaire à l'obtention d'un dimensionnement robuste et répondant aux objectifs de dimensionnement visés par les spécifications.

Une telle approche nécessite une prise de décision rapide lorsque les performances d'une architecture sont mesurées par rapport aux spécifications attendues. Son efficacité repose par conséquent sur l'utilisation de modèles de performances décrits par un jeu d'équations. Ce jeu d'équations pourra provenir soit d'une bibliothèque de circuits existants (réutilisation de modèles) soit d'un simulateur symbolique. L'approche par équations permet de piloter le processus de conception depuis la définition du système vers sa réalisation pratique. Elle favorise la descente des spécifications à travers les différents niveaux hiérarchiques depuis le niveau

système et la remontée des contraintes liées à la technologie et la topologie. L'utilisation d'équations permet donc de réduire le temps de calcul d'un dimensionnement à l'aide d'outils d'optimisation basée sur les équations. Contrairement à l'évaluation de performances obtenues par simulations, optimiser un jeu d'équations est en effet bien plus rapide car on a la possibilité d'adapter l'algorithme d'optimisation au type d'équations.

L'idée consiste à évaluer le plus rapidement les performances (même avec une certaine erreur) d'une topologie particulière. Le dimensionnement d'un circuit est accepté si l'évaluation de ces performances appartient au domaine des spécifications. Dans ce cas, il est possible de construire un sous-espace des ajustements dans lequel il existe au moins un point qui satisfasse l'ensemble des contraintes du problème de dimensionnement. Le problème de dimensionnement devient par conséquent d'abord un problème de satisfaction de contraintes (CSP). Cette approche prépare finalement l'étape de l'optimisation pour le dimensionnement en réduisant le nombre de degrés de libertés et en satisfaisant l'ensemble des spécifications et contraintes du jeu d'équations.

La connaissance de cet espace doit permettre d'une part la sélection de la topologie, et d'autre part d'utiliser cette connaissance pour quantifier la robustesse aux dérives de fabrication.

4.6.3 Mise en œuvre de notre approche

Résoudre un CSP nécessite de solutionner un problème inverse : connaissant les spécifications, nous cherchons à déterminer les ajustements. Pour répondre à cette problématique, nous proposons d'introduire une approche ensembliste. Une telle approche est particulièrement bien adaptée à notre objectif d'expliciter l'espace de conception. Elle est formulée de telle façon que l'espace de conception contienne à coup sûr la solution à notre problème de dimensionnement.

Mais construire un espace de conception par des formes non régulières telles que des ellipsoïdes rend une telle approche difficilement exploitable (au sens où elles ne facilitent pas le calcul de caractéristiques importantes tels que le volume, barycentre, …). Aussi, l'emploi d'intervalles de nombres réels permet une manipulation par les outils numériques plus adaptée pour le calcul. C'est à partir de cette formalisation particulière des ensembles qu'est fondé le concept de l'analyse par intervalles.

Ce concept permet avec un nombre limité d'opérations numériques de garantir l'encadrement des points images. Nous pouvons alors tester si un intervalle de l'espace des ajustements contient ou non au moins un point dont l'image projeté dans l'espace des performances est contenu dans les spécifications.
L'exploration de l'espace des ajustements permettra d'identifier le sous-espace des performances atteignables afin de guider le dimensionnement vers la bonne région. Avec cette approche, on construit une approximation de l'espace de solutions de façon à garantir l'existence de la solution optimale. Cette approche permet de capitaliser les analyses exploratoires en préparant à la fois à la sélection de topologie et à l'optimisation finale du dimensionnement de la topologie retenue.

Les avantages en analyse par intervalles permettent :

- de solutionner un problème inverse ;
- de donner une description complète de l'espace de conception ;

- de résoudre un problème d'optimisation globale même si l'espace est non convexe ou disjoint ;
- de rejeter immédiatement des cas sans solution ;
- de réajuster des spécifications trop sévères en aide à la sélection de topologies ;
- d'estimer la robustesse d'une solution architecturale au regard des variations du processus de production.

Cette nouvelle approche sera présentée en détail dans le prochain chapitre.

4.7 Conclusion

Ce chapitre a présenté les approches actuelles utilisées dans le dimensionnement d'architectures. Pour répondre aux contraintes de temps de conception, un processus automatique implémente en général des calculateurs à la recherche de la solution optimale d'une fonction de coût. Cette étape d'optimisation est particulièrement déterminante pour satisfaire les contraintes de conception d'un circuit et peut conduire à un processus coûteux en temps de calculs. Avec la complexité des problèmes numériques à résoudre, ce travail repose le plus généralement sur des outils d'aide à la conception comme les optimiseurs et deux approches se maintiennent : l'optimisation basée sur la simulation et l'optimisation basée sur l'équation. L'approche par simulation est très généraliste et demande peu d'effort pour analyser un circuit. En revanche, elle requiert un coût en terme de temps de calculs qui peut devenir très pénalisant pour le concepteur. L'approche par équations se base sur un jeu d'équations caractérisant les performances d'un circuit. La fonction de coût est explicitable et permet d'adapter l'algorithme d'optimisation au jeu d'équations. Il en résulte un gain de temps d'analyse qui facilite, en première approche, l'exploration de l'espace de conception. Cette facilité apporte à son tour davantage de visibilité de l'espace de conception.
En tirant les avantages de l'approche par équations, nous espérons d'une part, évaluer au plus vite la faisabilité de la solution architecturale, et d'autre part, proposer un premier dimensionnement de l'architecture retenue. Ce dimensionnement devra garantir une certaine robustesse face aux dérives de fabrication en construisant un espace de conception. Notre approche vise donc à préparer l'étape ultime d'optimisation en explicitant cet espace.

Une telle approche est mise en pratique à partir des concepts nés de l'analyse par intervalles et le prochain chapitre propose d'en présenter les méthodes.

Chapitre 5 - l'analyse par intervalles appliquée à l'exploration d'un espace

5.1 Introduction

Le dimensionnement d'un circuit analogique représente deux difficultés majeures : il requiert une part importante dans le temps de conception global d'une application et d'autre part le degré de liberté du problème de dimensionnement est très élevé. Le dernier point a traditionnellement été résolu comme un problème d'optimisation global multi-objectifs sous contraintes. La contrainte liée au premier point impose aujourd'hui de reconsidérer l'approche traditionnelle.

La recherche de la solution par des optimiseurs pose néanmoins d'autres difficultés telles que les problèmes de convergence des algorithmes, la présence de minima locaux, le choix des conditions initiales, etc. Néanmoins, tous les optimiseurs nécessitent un préalable à la recherche de minima, c'est la satisfaction des conditions imposées par les contraintes : le CSP (*Constraint Satisfaction Problem*). L'ensemble des points vérifiant les contraintes appartiennent à l'ensemble « faisable ». L'analyse de cet ensemble permet d'apporter rapidement des preuves numériques d'absence ou d'existence de points solutions. Son explicitation formelle permet de mener une recherche exploratoire de l'espace de conception en obtenant l'ensemble des points candidats à la minimisation. Cela offre une alternative au « front de Pareto » qui ne décrit qu'un bord de cet ensemble faisable. La connaissance de son « volume », nous permettra d'introduire au chapitre 6 une métrique de la robustesse d'une architecture de circuit.

Mais expliciter « l'ensemble des points solutions » est un problème inverse qui est extrêmement difficile voire impossible à résoudre avec les méthodes classiques de l'analyse numérique (méthode ponctuelle). Par opposition, les méthodes ensemblistes permettent une résolution rigoureuse d'une grande classe de problèmes non-linéaires en optimisation globale [Han-03]. La mise en œuvre du calcul ensembliste est assurée par les méthodes de l'analyse par intervalles. De telles méthodes permettent d'analyser rapidement de larges portions de l'espace à explorer en garantissant le résultat obtenu. Appliquées au dimensionnement d'une topologie de circuit électrique (aussi appelé architecture de circuit), ces méthodes ensemblistes retournent alors un encadrement de l'espace « faisable » des ajustements. En contrepartie, les intervalles calculés s'accompagnent d'une surestimation de la région analysée. Pour réduire cet excès, diverses techniques ont été proposées et mettent en balance leur efficacité en terme de temps de calculs et leur résolution dans l'approximation. Ce chapitre présente uniquement les quelques techniques que nous avons retenues en fonction de ces deux critères de performance.
La partie 2 de ce chapitre introduit les notions de bases en analyse par intervalles. Ces notions seront indispensables au développement des approches utilisées dans le cadre de l'exploration de l'espace de conception. La partie 3 présente la représentation ensembliste par pavage utile à l'approximation d'un ensemble solution. La partie 4 mettra en œuvre la représentation par pavage à l'aide de l'algorithme d'inversion ensembliste. Dans la partie 5, une classe particulière d'opérateurs connus sous le nom de « contracteurs » sera proposée. Ces opérateurs sont au cœur de notre approche et nous montrerons comment nous les utiliserons en conception (en dimensionnement d'un circuit) dans la partie 6.

Remarque : Nous introduirons l'arithmétique par intervalles en utilisant les espaces **A** et **P** en référence au paragraphe 4.2 du chapitre 4.

5.2 L'analyse par intervalles

Avant de montrer comment ce nouveau type de formalisme s'applique à l'exploration d'espace des ajustements, nous allons tout d'abord rappeler quelques définitions et les objectifs qui sous-tendent à la méthode de dimensionnement.

Définition d'un intervalle
Un intervalle $[a]$ est un sous-ensemble connexe fermé et borné de l'ensemble des réels **R**

$$[a] \triangleq \{a \in \mathbf{R} / a^- < a < a^+\} = [a^-, a^+], \text{avec } (a^-, a^+) \in \mathbf{R}^2$$

avec, a^- = min$[a]$ et a^+ = max$[a]$ respectivement la borne inférieure et la borne supérieure. L'ensemble de tous les intervalles fermés dans **R** sera noté par **IR** et finalement $[a] \in$ **IR**.

On peut ensuite généraliser cette définition à l'espace $\mathbf{R}^n$ avec l'écriture vectorielle, ainsi l'intervalle dans **IR** devient un pavé dans $\mathbf{IR}^n$ ce qui justifie l'écriture vectorielle.

Définition d'un pavé
Un pavé se définit par le produit cartésien d'intervalles réels :

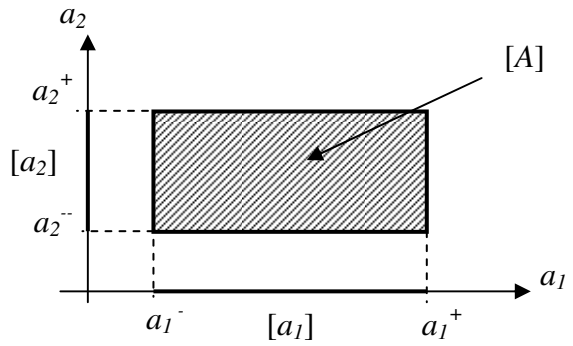
$$[A] \triangleq [a^-_1, a^+_1] \times \ldots \times [a^-_n, a^+_n] = \left[[a^-_1, a^+_1], \ldots, [a^-_n, a^+_n] \right]^T, \text{où } [A] \in \mathbf{IR}^n$$

On définit alors un pavé de $\mathbf{IR}^n$ ou vecteur intervalle de $\mathbf{IR}^n$ comme étant le produit cartésien de n intervalles de **IR**.

Le produit cartésien dans **R** permet d'introduire une interprétation géométrique d'un pavé. Par exemple, un pavé $[A]$ appartenant à $\mathbf{IR}^2$ défini par :

$$[A] \triangleq [a_1] \times [a_2] = [a^-_1, a^+_1] \times [a^-_2, a^+_2]$$

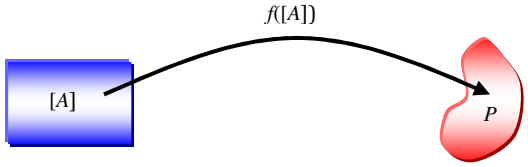
représente un pavé rectangulaire (plus simplement appelé un rectangle) dans le plan $\mathbf{R}^2$:



Regardons maintenant comment calculer l'image par une fonction f d'un pavé. Plusieurs observations s'imposent :

- Première observation : le calcul de l'image d'un pavé par une méthode ponctuelle est difficile ; on ne peut calculer qu'un nombre fini de points. Or un intervalle forme un ensemble dont le nombre de points est infini. Avec des hypothèses de continuité dans les voisinages (petites variations), on peut admettre « deviner » l'ensemble complet.

- Seconde observation : le calcul de l'image d'un pavé ne correspond pas, en général, à un pavé :



Si f est continue, alors la seule propriété dont on dispose est que si $[A]$ est compact alors P est aussi un compact mais pas nécessairement un pavé.

De ces deux observations nous pouvons déduire l'objectif que nous assignons aux calculs par intervalles. On souhaite donner une approximation de l'ensemble image P garantissant que tous les points de P sont inclus dans l'approximation.
L'approximation peut avoir une forme géométrique quelconque : polytope, zonotope, ellipsoïde, hypercube, etc [Rei-99]. Pour réduire au maximum les calculs numériques, nous choisissons d'approximer un ensemble P par un pavé $[P]$. Ceci induira comme contrepartie une « sur approximation » optimiste.
Calculer $[P]$ à partir d'un pavé $[A]$ pour une fonction f nécessite l'introduction de la notion de fonction d'inclusion.

- Définition de la fonction d'inclusion :

La fonction d'inclusion recherche un pavé qui contienne de façon garantie l'image du pavé [A] par la fonction f. La fonction d'inclusion est telle que l'image d'un pavé est aussi un pavé. La fonction d'inclusion permet donc de conserver les propriétés de compacité et de convexité du pavé [A] dans l'espace image.

Considérons une fonction réelle f : **R**→**R**, la fonction $[f]$: **IR**→**IR** sera appelée fonction d'inclusion de f et elle vérifiera pour tout intervalle $[a]$ le théorème fondamental de Moore:

$f([a]) \subseteq [f]([a])$ [Moo-79]

Remarque : Aucune hypothèse sur f n'est nécessaire pour la définition d'une fonction d'inclusion.

La figure 5-1 illustre le cas où l'image d'un pavé $[A]$ par la fonction f est contenue dans un pavé $[P]$. On note alors $[P] \equiv [f]([A])$ ce pavé que l'on appellera « *pavé englobant* »8 ou « *Enclosing Box* ».

8 Le terme englobant est plus générique et sera préféré au terme encadrant qui ne s'adresse qu'aux cas de pavés dans un espace à deux dimensions.

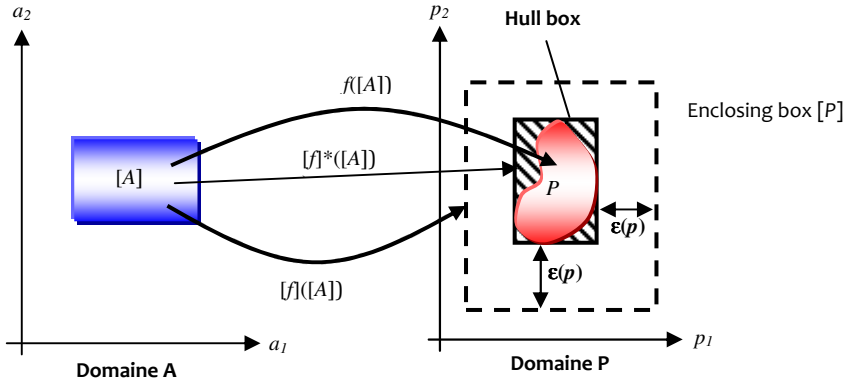


Figure 5-1 Image directe d'un pavé de l'espace des ajustements dans l'espace des performances par une fonction f.

Le conteneur qui en résulte est construit avec une surestimation par rapport à l'enveloppe réelle du sous-espace $P = f([A])$. Cette surestimation est notée $\varepsilon(p)$ sur la figure 5-1.

On peut généraliser l'écriture précédente à un espace de dimension supérieure en considérant une fonction f de R^n à R^m pour laquelle il existe m fonctions d'inclusion $[f_{j=1,\dots,m}]$ associées aux fonctions composantes f_j de f. Le produit cartésien de chacune de ces fonctions d'inclusion permet alors de définir la fonction d'inclusion de f par :

$$[f]([A]) = [f_1]([A]) \times \dots \times [f_m]([A])$$

- Définition de la fonction minimale

La fonction d'inclusion $[f]^*$: IR→IR sera appelée fonction d'inclusion minimale de f si elle vérifie pour tout intervalle $[a] \in IR^n$:

$$[f]^*([a]) \triangleq [f(a)] \subseteq [f]([a])$$

Remarque : La fonction d'inclusion d'une fonction monotone (à dérivée de signe constant) sera minimale.

Une fonction minimale génère un pavé de dimension minimale c'est-à-dire que chacune des faces du pavé est en contact avec un point extremum de l'ensemble P. Ce conteneur particulier est appelé *Hull box* et est défini par:

$$[f_i]^*([a_i]) = [p_i] = [p_i^-, p_i^+]$$
où : $p_i^- = min(f_i([a_i]))$ et $p_i^+ = max(f_i([a_i]))$ pour $i = 1,\dots,n$

Le pavé minimal vérifiera donc toujours la relation d'inclusion :

$$f([a]) \subseteq \left[f([a]) \right] = [f]^*([A]) \subseteq [f]([A])$$

La figure 5-1 représente [*f*]*([*A*]) par une zone hachurée. Pour tout ensemble *P*, le conteneur *Hull box* est unique9.
Calculer le pavé minimal en recherchant les extrema d'un ensemble par des méthodes d'analyses numériques sera coûteux en temps de calculs. L'approche par un pavé sera donc extrêmement économique en temps de calcul. Néanmoins, il existe alors une infinité de pavés qui approximent avec plus ou moins d'efficacité cet ensemble.
Si l'on accepte de ne pas rechercher LE pavé minimal, alors on doit accepter qu'il existe plusieurs fonctions d'inclusions, chacune générant un pavé englobant.

Construction d'une fonction d'inclusion :

Comme nous l'avons souligné ci-dessus, il existe plusieurs principes pour construire une fonction d'inclusion. La façon la plus simple et « naturelle » est de remplacer chacune des variables a_i de la fonction $\boldsymbol{f}$ par son intervalle la contenant [a_i] et chaque opérateur arithmétique ou fonction élémentaire par son équivalent dans **IR**. On dira alors que [$\boldsymbol{f}$] est la fonction d'inclusion naturelle de $\boldsymbol{f}$. [Jau-01b, p27] a montré que si les pavés sont de grandes tailles, cette fonction d'inclusion donnera la meilleure approximation.
Si au cours du processus de raffinement de l'approximation, les pavés deviennent plus petits, alors d'autres fonctions d'inclusions telles que : (*Mean value Inclusion Function, Centred Inclusion Function, Taylor Inclusion Function, Newton Inclusion Function, etc*) [Han-03, Jau-01a, p33] peuvent être utilisées.

Discutons de l'origine et du « contrôle » de la surestimation.

La surestimation a deux origines. La première origine est la conséquence de l'utilisation d'un pavé pour englober un ensemble quelconque. La forme géométrique d'une telle enveloppe introduit par nature une surestimation connue sous le nom de « w*rapping Effect* ». Etant donné que l'utilisation de pavés facilite les calculs, nous ne reviendrons donc pas sur son choix. La seconde origine est liée au « phénomène de dépendance » ou « *depedency effect* » [Jau-01b] entre les variables de la fonction.

En effet, la présence de multiples occurrences d'une ou plusieurs variables au sein d'une équation entraîne une surestimation. On obtiendra une minimisation de cette surestimation si chaque variable a_i apparaît de manière unique dans l'expression de $\boldsymbol{f}(\boldsymbol{a})$.

Pour réduire la surestimation produite par les fonctions d'inclusions, d'autres techniques existent. On peut pré-conditionner les expressions explicites des équations en réduisant l'occurrence des variables. La factorisation est un exemple de pré conditionnement. D'autres techniques, issue des travaux de l'intelligence artificielle, traite le problème de la surestimation par les techniques de « contraction d'intervalles » [Jau-01b]. Les techniques de contraction s'appuient sur des opérateurs qui ont en charge de réduire des pavés et que l'on appelle des « contracteurs ». Ceux-ci permettent de minimiser la surestimation d'une fonction d'inclusion naturelle même en présence de multi occurrence de variables.

Compte tenu des arguments développés ci-dessus, nous choisissons pour la suite de la thèse les fonctions d'inclusion naturelle avec la technique de réduction des surestimations par des opérateurs de contraction. Mais avant de développer les techniques de contraction, nous allons

9 Ceci est vrai par approximation « externe » (*Wrapping*) du sous-espace *P* mais la même approche par approximation « interne » ne peut pas retourner un pavé minimal unique !

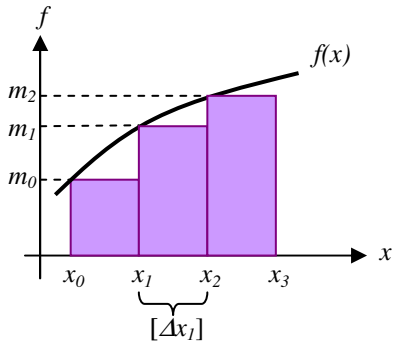
d'abord nous intéresser à la façon dont il est possible d'expliciter un ensemble quelconque et la zone d'incertitude qui accompagne son approximation.

5.3 Illustration des principes de recouvrement et d'encadrement

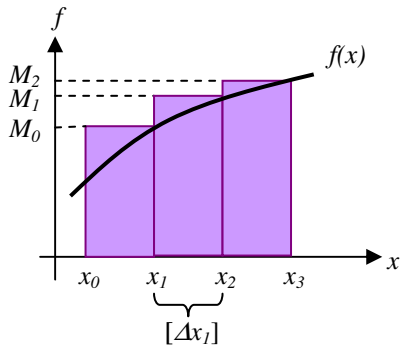
Nous allons, dans cette partie, montrer comment le concept de recouvrement peut être appliqué pour expliciter un ensemble quelconque et détailler le principe de son approximation.

Le concept de recouvrement est un concept fondamental en mathématiques. Il est par exemple mis en œuvre dans le calcul intégral au sens de Riemann en cherchant à évaluer l'aire « sous une courbe ». L'intégration s'appuie sur le découpage d'un espace borné en une succession de petits intervalles conduisant à la juxtaposition de pavés image (ici des rectangles) représentant une fonction « en escalier » dont l'aire sous la courbe est égale à la somme de l'aire des rectangles.

Pour illustrer ce concept, considérons, sur la figure ci-dessous, la fonction « en escalier » choisie constante sur des intervalles $[\Delta x_i] = [x_i , x_{i+1}]$ réguliers.



Recouvrement minorant



Recouvrement majorant

Le domaine sous la courbe de la fonction $f(x)$ peut alors être vu comme une réunion de pavés dont une dimension est donnée par la largeur de l'intervalle $w([\Delta x_i])$ et l'autre dimension par la largeur de l'intervalle image $w([0 , m_i])$. L'aire sous la fonction $f(x)$ s'exprime par :

$$\int_{x_0}^{x_3} f(x).dx > \sum_{i=0}^{i=2} m_i.w\left([\Delta x_i]\right)$$

L'intégration est ici minorante ; elle donne une approximation par un recouvrement minorant de l'aire sous la fonction $f(x)$:

$$\bigcup_{i=0}^{2}\left([m_i],[\Delta x_i]\right) \subset Aire\left(f(x),[x_0,x_3]\right)$$

Mais l'intégration de la courbe peut également se calculer par la somme des rectangles situés sur la courbe $f(x)$. Dans ce cas :

$$\int_{x_0}^{x_3} f(x).dx < \sum_{i=0}^{i=2} M_i.w\left([\Delta x_i]\right)$$

L'intégration est alors majorante ; elle donne une approximation par un recouvrement majorant de l'aire sur la fonction *f(x)* :

$$Aire\left(f(x),[x_0,x_3]\right) \subset \bigcup_{i=0}^{2}\left([M_i],[\Delta x_i]\right)$$

L'aire de la courbe peut être finalement encadrée par ces deux recouvrements minorant et majorant :

$$\bigcup_{i=0}^{2}\left([m_i],[\Delta x_i]\right) \subset Aire\left(f(x),[x_0,x_3]\right) \subset \bigcup_{i=0}^{2}\left([M_i],[\Delta x_i]\right)$$

Cet encadrement permet finalement de « contenir » la surface recherchée. Un tel concept illustre parfaitement la méthodologie qui est la notre pour expliciter un sous-espace par l'encadrement par deux recouvrements.

Le concept de recouvrement que nous avons choisi met en œuvre la représentation d'un sous-espace par pavage conduisant à une approximation minorante ou majorante de l'espace de recherche. Nous commencerons par introduire ce concept de recouvrement par la représentation d'un sous-espace par pavage. Nous montrerons ensuite comment est appliqué ce concept à l'encadrement du sous-espace recherché.

5.3.1 Description d'un sous-espace par la méthode de recouvrement

L'idée de base de la représentation ensembliste par recouvrement consiste à décomposer un pavé de l'espace **A** en une réunion de sous-pavés de l'espace **A**. La réunion de ces sous-pavés forme alors une famille de pavés ayant à la fois des propriétés de compacité, de convexité et de connexité.

Le pavage de l'espace peut être effectué selon deux techniques. Soit sous la forme d'un pré découpage systématique de manière équivalente à l'application d'une grille (*griding*). Soit sous la forme d'un découpage itératif. Pour commencer, nous allons d'abord considérer le cas général d'un pré découpage systématique.

Considérons le problème suivant : soit un ensemble quelconque *A* et soit un pavé de recherche $[A]_o$, nous souhaitons donner une description explicite de l'espace A_S formée par l'intersection de ces deux sous espaces : $A_S = A \cap [A]_o$.
Dans la figure 5-2 on illustre le cas où $A \subset [A]_o$, c'est à dire que l'on aura aussi $A_S = A$. On y observe l'approximation par pavage du sous-espace A_S.

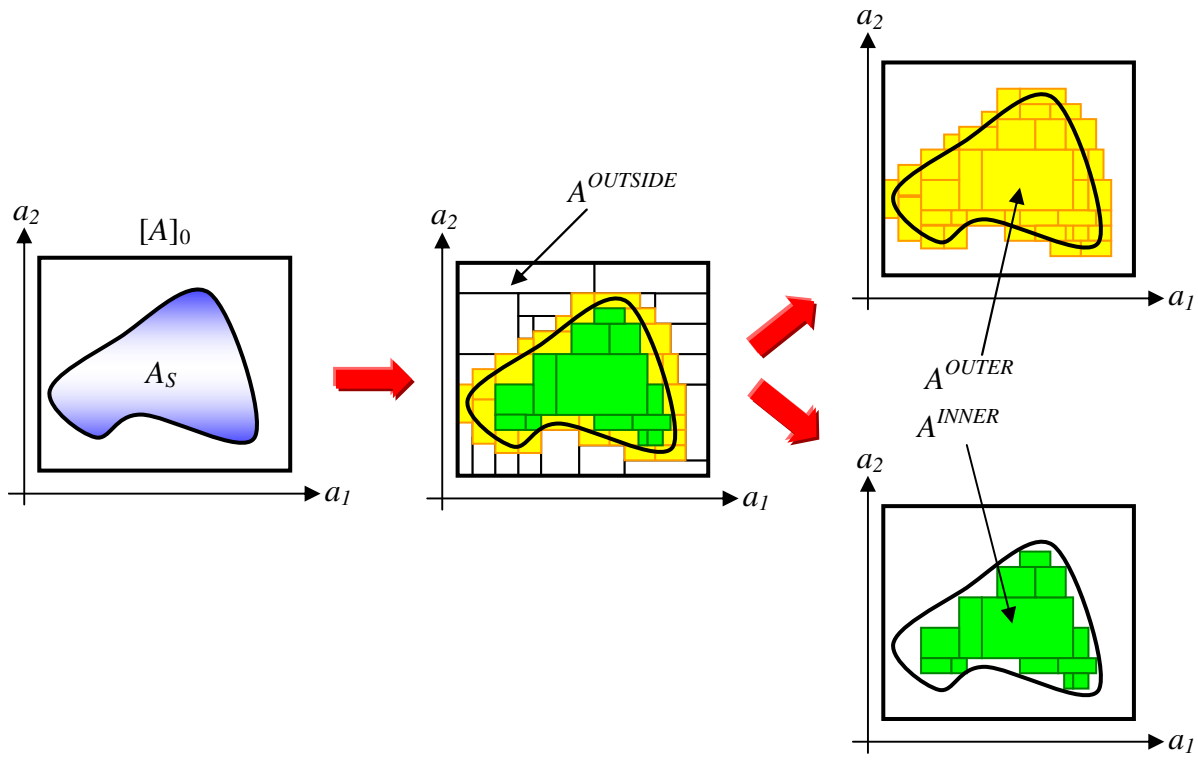


Figure 5-2 Approximation du sous-espace A_S par pavage

Nous nommerons A^{OUTER} l'ensemble des pavés $[A]_i$ dont la réunion forme une surestimation de l'ensemble A_S et que nous exprimons par :

$$\left\{ [A]_i \,/\, \bigcup_{i=1}^{N} [A]_i \cap A_S \neq \varnothing \right\} \triangleq A^{OUTER}$$

De la même façon, il est possible de donner une estimation sous approximante de A_S. Nous nommerons A^{INNER} le sous-espace représenté par la réunion des pavés $[A]_i$ inclus dans A_S et que nous exprimons par :

$$\left\{ [A]_i \,/\, \bigcup_{i=1}^{n} [A]_i \subset A_S \right\} \triangleq A^{INNER}$$

Cette double représentation à la fois « majorante » et « minorante » de l'ensemble A_S, permet d'approcher la frontière du sous-espace A_S. Nous pouvons dans ce cas définir une nouvelle classe de pavés qui formeront le sous espace A^{BORDER}. L'ensemble des pavés $[A]_i$ appartenant à A^{BORDER} regroupe les pavés qui appartiennent au sous espace A^{OUTER} mais ne font pas partie de A^{INNER} :

$$\left\{ [A]_i \,/\, \bigcup_{i=1}^{n} [A]_i \cap A_S \neq \varnothing \wedge \bigcup_{i=1}^{n} [A]_i \not\subset A_S \right\} \triangleq A^{BORDER}$$

La réunion des pavés de la classe A^{BORDER} représente le domaine contenant la frontière du sous-espace A_S.

De ces définitions, on peut déduire deux nouvelles relations entre ces trois sous espaces :

$$A^{OUTER} \triangleq A^{BORDER} \cup A^{INNER}$$
$$A^{INNER} \subset A_S \subset A^{OUTER}$$

Enfin, un dernier sous espace peut être défini. C'est le sous espace complémentaire au sous espace A^{OUTER} est que nous nommerons $A^{OUTSIDE}$. Ce sous espace ne contient aucun point de l'ensemble A_S et il sera défini par :

$$A^{OUTSIDE} \triangleq \left\{ [A]_i \,/\, [A]_i \cap A_S = \varnothing \right\}$$

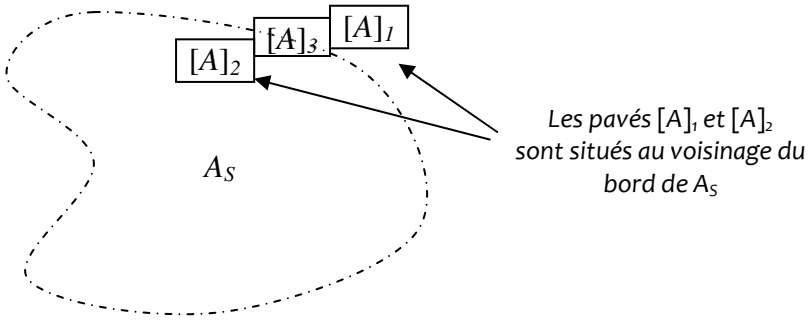
Nous disposons finalement de trois sous-espaces complémentaires, c'est à dire des sous-espaces dont les pavés sont dissociés. Si un pavé appartient au sous espace A^{INNER} alors ce même pavé ne peut pas appartenir aux sous espaces A^{BORDER} ou $A^{OUTSIDE}$. De ce constat nous introduisons la notion de classe d'appartenance d'un pavé $[A]_i$.

Dans nos applications, nous travaillerons avec des algorithmes qui interpréteront la notion de sous-espace par une liste pavé d'une même classe. Cette notion de liste de pavés se formalise pour une liste de *n* pavés de la manière suivante :

$$\underset{i=1}{\overset{n}{L}}{}^{X}\left([A]_i\right) \triangleq \left\{ [A]_1, [A]_2, ..., [A]_n \right\},$$

qui est la liste nommée *X* contenant *n* pavés différents $[A]_i$.

Précisons qu'un sous-espace est décrit par un recouvrement majorant. La figure ci-dessous illustre ce cas par le recouvrement partiel de la frontière d'un espace A_S.



Le recouvrement majorant de la frontière est alors donné par la liste qui contient les trois pavés de bord $[A]_1$, $[A]_2$, et $[A]_3$:

$$\underset{i=1}{\overset{3}{L}}{}^{BORDER}\left([A]_i\right) \triangleq \left\{ [A]_1, [A]_2, [A]_3 \right\}$$

Décrire un sous-espace c'est donc donner la liste des pavés du recouvrement contenant la frontière tel que :

$$A_S \subset \bigcup_{i=1}^{3} [A]_i = A^{BORDER}$$

Mais compte tenu de la surestimation par la fonction d'inclusion $[f]$ et de la « proximité » des pavés $[A]_1$ *et* $[A]_2$ de la frontière de A_S, l'opération de classification ne peut pas affirmer :

- qu'il existe un point de $[A]_1$ appartenant à A_S c'est-à-dire recouvrant sa frontière ; ce pavé ne peut donc pas être classé dans $A^{OUTSIDE}$,
- qu'il existe un point de $[A]_2$ n'appartenant pas à A_S ; ce pavé ne peut donc pas être classé dans A^{INNER}.

Pour cette raison, chaque pavé de la liste L^{BORDER} pris séparément sera appelé indéterminé par l'algorithme de classement. Ce constat conduit à la remarque suivante :

On peut affirmer que l'union de tous les pavés indéterminés formée par A^{BORDER} contient la frontière du sous-espace A_S, mais on ne peut pas affirmer qu'un pavé de cette union contienne de façon sûre cette frontière.

5.3.2 Cas particulier de la description d'un sous-espace

Au contraire du problème traité précédemment, ici l'ensemble A et le pavé de recherche $[A]_0$ ne vérifie pas l'hypothèse : $A \subset [A]_0$. La figure 5-3 illustre ce cas particulier et met en évidence les trois classes : A^{INNER}, A^{BORDER}, $A^{OUTSIDE}$ avec la particularité d'obtenir des pavés de la classe A^{INNER} en bordure du pavé de départ $[A]_0$.

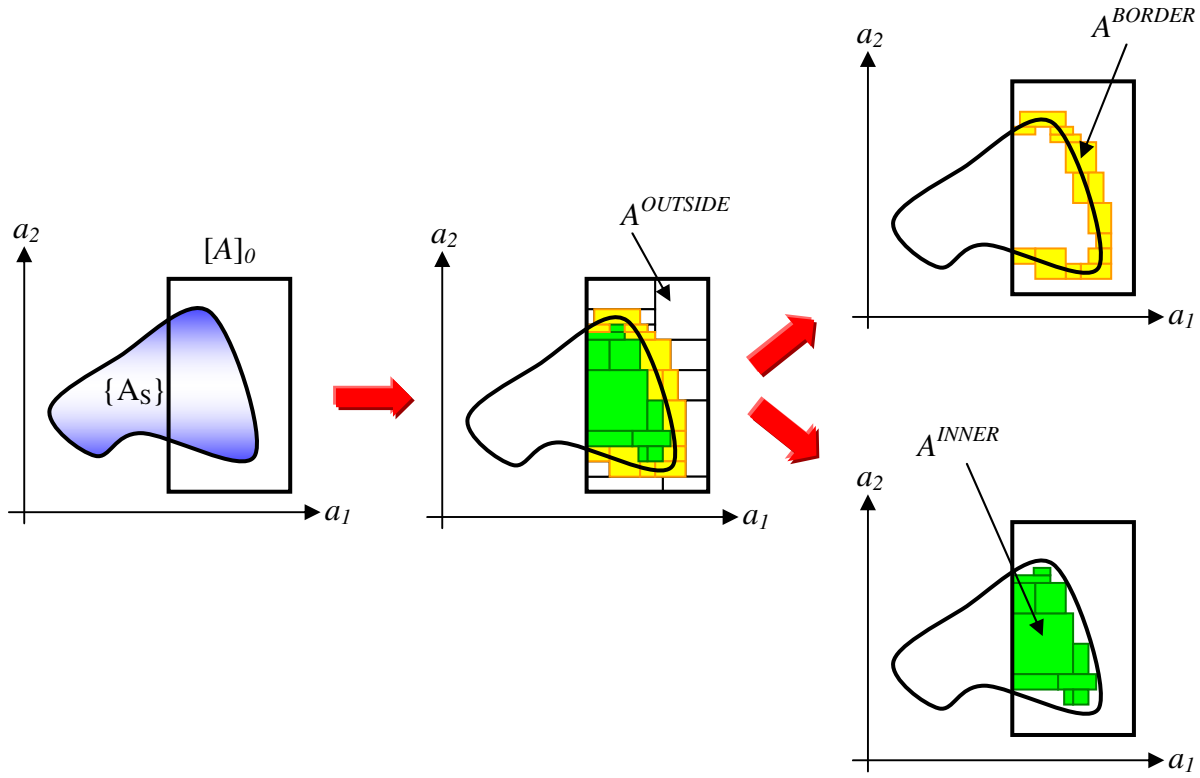


Figure 5-3 Approximation partielle du sous-espace A_S par pavage

Les exemples traités jusque là ont été illustrés à partir d'un pavage de $[A]_0$ déterminé par pré découpage systématique. Cette technique pose la question de l'efficacité d'un tel processus de découpage très consommateur en opération de division. Une solution alternative est alors de procéder à une division uniquement si nécessaire. La division se fera donc au sein d'un processus itératif.

5.3.3 Le pavage par division itérative

Le pavage par division itérative s'appuie sur la propriété du calcul par intervalles qui précise que lorsque la largeur d'un intervalle tend vers zéro, la largeur de l'intervalle image tend également vers zéro [Jau-01b, p28].
La phase de division consiste à partager un pavé en plusieurs sous-pavés contigus mais sans recouvrement. Si la phase de division consiste à partager un pavé en deux sous-pavés, on parlera de bissection ; si la division conduit à trois sous-pavés, on parlera de trisection. Dans ce chapitre et pour des raisons de simplicité, nous limitons, le découpage au partage par bissection.
La division n'affecte que les sous-pavés contenus dans la classe A^{BORDER} car ce sont les seuls pavés pour lesquels l'on n'a pas de certitude sur leur contenu : tous les points sont-ils dans A_S ou sont-ils en-dehors de A_S ? Le processus itératif de division se poursuivra jusqu'à atteindre le niveau d'approximation acceptable pour la description explicite de l'ensemble solution. Le critère d'arrêt du processus sera la largeur maximale d'un pavé avant division. Si la plus grande largeur d'un pavé passe sous le seuil (*threshold*) on estime qu'il tombe dans l'incertitude de mesure. Le processus itératif sera interrompu lorsque la taille de tous les pavés de la classe A^{BORDER} sera inférieure à un seuil correspondant par exemple aux tolérances accompagnant les contraintes. Ce seuil est arbitraire et permet de contrôler la largeur de la zone d'incertitude qui contient la frontière. Il vient naturellement que plus le nombre d'opérations de bissection sera élevé, plus faible sera alors l'erreur d'approximation du sous-espace solution recherché.

Le processus est résumé ci-dessous pour une itération :

```
Let a list L_{j=1}^{m}([A_j]) of interval [A]_j,
    if dim(L)>0
            for j=1 to m
                [A]_j = L(j)
                if width([A]_j) > threshold
                    Bisect [A]_j ⇒ [A]_j1 ∪ [A]_j2,
                    Put [A]_j1, [A]_j2 ⇒ L*
                end if
            end for
            L* ⇒ L
        end if

End
```

Notons qu'un tel processus nécessite des ressources en calculs qui augmentent avec la dimension du problème à traiter. En effet, en considérant que pour chaque itération, une opération de bissection est pratiquée sur chacune des dimensions de l'espace d'ajustement, il en résulte alors qu'un pavé est le produit de 2^n sous-pavés où n représente la dimension de l'espace des ajustements. Ainsi après k itérations, le nombre total de sous-pavés s'élève à $2^{n.k}$ rendant le travail de classification extrêmement fastidieux. Cette approche génère un nombre de pavés qui devient très vite important lorsque la précision recherchée augmente. Ce problème est une véritable limitation à la conception des circuits analogiques surtout lorsque les dimensions à traiter sont supérieures à 10 [Jau-01a]. Nous verrons au paragraphe 5.5, qu'une alternative intéressante est alors le recours aux techniques de contraction d'espace des ajustements qui

permettent d'éliminer rapidement de large portion de l'espace d'ajustement avant toute opération de division.

Nous avons présenté le principe de description explicite d'un ensemble (ou sous espace) A_S qui repose sur la classification des sous-pavés. Notons que la détermination de ces classes suppose précisément que A_S soit connu ! En revanche, aborder le travail de classement de pavés lorsque A_S est inconnu pose le problème du calcul de l'image inverse. Nous allons donc voir comment les notions de pavage peuvent être appliquées à la caractérisation du sous-espace A_S en abordant l'approche par inversion ensembliste.

5.4 Description d'un sous-espace par la méthode d'encadrement

L'approche par inversion ensembliste consiste à caractériser un ensemble solution A_S de **A** en considérant un pavé [*S*] dans **P** tel que :

$$A_S = f^{-1}([S])$$

Rechercher le sous-espace A_S revient à résoudre un problème inverse qui est extrêmement difficile avec les techniques ponctuelles classiques et la recherche de A_S est impossible sans la connaissance explicite de la fonction inverse f^{-1}. Cela est en revanche possible à l'aide des techniques de l'analyse par intervalles en s'appuyant d'une part sur la représentation par pavage et d'autre part en considérant le calcul direct de l'image d'un pavé :

la représentation par pavage consiste à encadrer A_S par deux sous-espaces $\boldsymbol{A}^{INNER}$ et $\boldsymbol{A}^{OUTER}$ tels que :

$$\boldsymbol{A}^{INNER} \subset A_S \subset \boldsymbol{A}^{OUTER};$$

le calcul direct de l'image d'un pavé $[A]_i$ utilise les fonctions d'inclusions pour calculer le pavé image [*P*] ; sa comparaison au pavé [*S*] permet de procéder à une « rétro classification » du pavé $[A]_i$ dans l'une des trois classes.

Cette approche a été appliquée dans l'algorithme SIVIA (*Set Inversion Via Interval Analysis*) [Jau-94]. Nous en présentons les concepts de base.

5.4.1 Concepts de base de l'inversion ensembliste

L'approche de l'inversion ensembliste repose sur le concept de fonctions d'inclusions pour exprimer l'image d'un pavé [*P*]=[*f*]([*A*]). Selon le théorème fondamentale de Moore on sait que l'on vérifie la relation suivante : $f([a]) \subseteq [f]([a])$ [Moo-79]. Par conséquent, on a la garantie que tout point du pavé [A] à son image dans le pavé [*P*].
L'objectif étant de vérifier la condition [*P*] = [*S*]. Nous pourrons alors statuer sur l'impact d'un pavé [*A*] sur cette condition. Si ce pavé [*P*] est à l'extérieur du pavé des spécifications [S], alors tous les points de ce pavé [*A*] appartiennent au sous espace $A^{OUTSIDE}$ et ce pavé appartient à la classe $A^{OUTSIDE}$. Le scénario est illustré sur la figure 5-4 :

Si $[P] \cap [S] = \varnothing$ alors $A_S \not\subset [A]$ (figure 5-4a)

A l'opposé, si le pavé [*P*] est totalement inclus dans le pavé des spécifications [*S*], alors on a la garantie que tous les points du pavé [*A*] ont les images dans le pavé des spécifications et ce pavé [*A*] appartiennent au sous espace A^{INNER}. Nous traduisons ce cas par :

Si $[P] \subset [S]$ alors $[A] \subset A_S$ (figure 5-4b)

Si le pavé [A] ne satisfait aucun des cas précédents, alors il appartient à la classe A^{BORDER} et vérifie les relations suivantes :

Si $[P] \cap [S] = \varnothing$ alors $\wedge [P] \not\subset [S]$ (figures 5-4c et 5-4d)

Un pavé de la classe A^{BORDER} est souvent appelle « *Undertermined* » car il n'offre pas de réponse tranchée sur l'appartenance de l'ensemble de ces point à l'ensemble A_S.

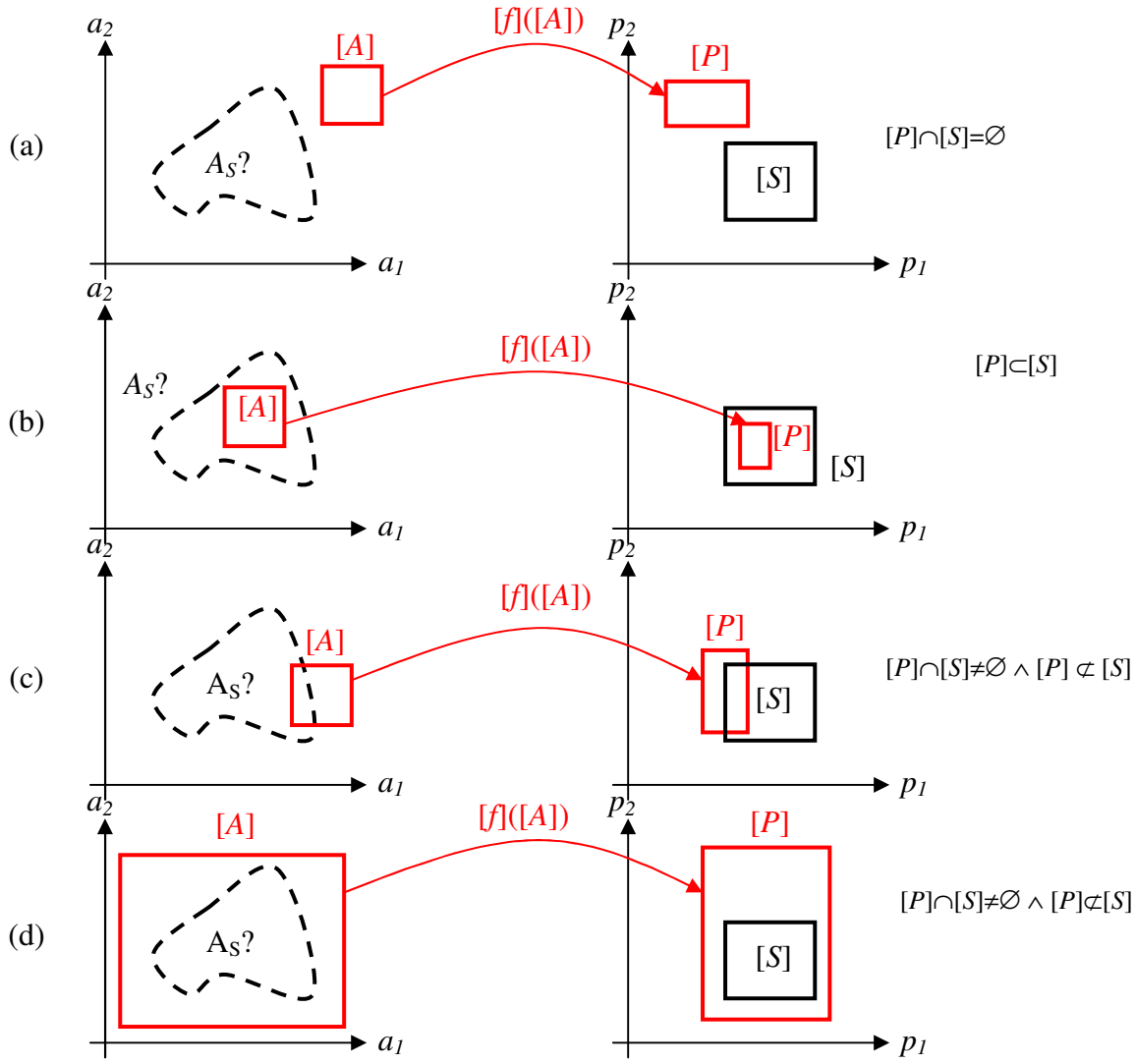


Figure 5-4 Test de pavés dans l'espace P

Les processus de description d'un ensemble par sous pavages de l'espace et d'inversion ensembliste par classification des pavés ont été introduit avec la technique de pré découpage systématique. Nous allons présenter maintenant l'inversion ensembliste en utilisant la division itérative.

5.4.2 Illustration du concept de l'inversion ensembliste par division itérative

Dans cet exemple, nous exposons un autre cas de figure qui est courant en conception. Nous ne connaissons pas le sous-espace A_S mais choisissons arbitrairement un pavé $[A]_0$ dans lequel nous rechercherons les solutions partielles aux problèmes du CSP.
Le processus de division utilisé sera ici itératif mais peut naturellement démarrer avec un pavé plus large englobant A_S comme dans le cas précédent.
Considérons un intervalle $[S]$ exprimé dans l'espace **P** que nous limitons à une dimension. L'intervalle $[S]$ correspond à un sous-espace A_S représenté dans un plan de l'espace **A** à deux dimensions. Une illustration de cette approche est donnée figure 5-5.

En partant du pavé $[A]_0$, on évalue son image $[f][A]_0 = [P]$ dans l'espace de performances. Le pavé $[A]_0$ est tracé en rouge. En comparant le pavé $[P]$ au pavé des spécifications $[S]$, il vient que seuls certains points performances sont inclus dans l'intervalle des spécifications. Le pavé $[A]_0$ est donc classé A^{BORDER} et du fait de son classement, ce pavé est bissecté en deux sous-pavés symétriques $[A]_{00}$ et $[A]_{01}$. Le sous espace A^{BORDER} est alors décrit par la liste $L^{BORDER} = \{[A]_{00},[A]_{01}\}$ contenant l'ensemble des pavés dont la réunion forme le sous espace A^{BORDER}.

Après la première bissection, une nouvelle itération démarre. Durant cette seconde itération, on évalue les deux pavés images $[P]_{00}$ et $[P]_{01}$. On opère le classement de ces deux pavés. Le pavé $[A]_{00}$ vérifie la relation suivante : $[f]([A]_{00}) \cap [S] = \varnothing$; le sous-pavé $[A]_{00}$ est donc classé $A^{OUTSIDE}$ et ne sera ni bissecté ni retraité. En revanche, le pavé $[A]_{00}$ est classé A^{BORDER} et sera bissecté si sa taille est supérieure au seuil. Après bissection, la liste des pavés de bord se résume donc à : $L^{BORDER} = \{[A]_{001},[A]_{012}\}$.

Lors de la troisième itération le pavé $[A]_{012}$ vérifie $[f]([A]_{012}) \subset [S]$ faisant de $[A]_{012}$ un sous-pavé classé dans la liste des pavés intérieurs $L^{INNER} = \{[A]_{012}\}$. Ce sous-pavé sera tracé d'un rectangle de fond vert sur la figure 5-5c. Le sous-pavé $[A]_{011}$ est quant à lui est classé A^{BORDER}. Il sera donc bissecté en $[A]_{0111}$ et $[A]_{0112}$ si sa taille est toujours supérieure au seuil. A la fin de cette troisième itération, les listes sont : $L^{OUTSIDE} = \{[A]_{00}\}$, $L^{BORDER} = \{[A]_{0111}, [A]_{0112}\}$ et $L^{INNER} = \{[A]_{012}\}$. Le processus aboutit à la représentation donnée figure 5-5e où le sous-espace des pavés intérieurs est constitué par la réunion de l'ensemble des sous-pavés de la liste L^{INNER} lui-même délimité par la réunion des pavés de bords de la liste L^{BORDER}.

Le processus itératif s'arrêtera lorsque soit tous les pavés de la liste L^{BORDER} auront une taille inférieure au seuil de bissection, soit que la liste L^{BORDER} sera vide.

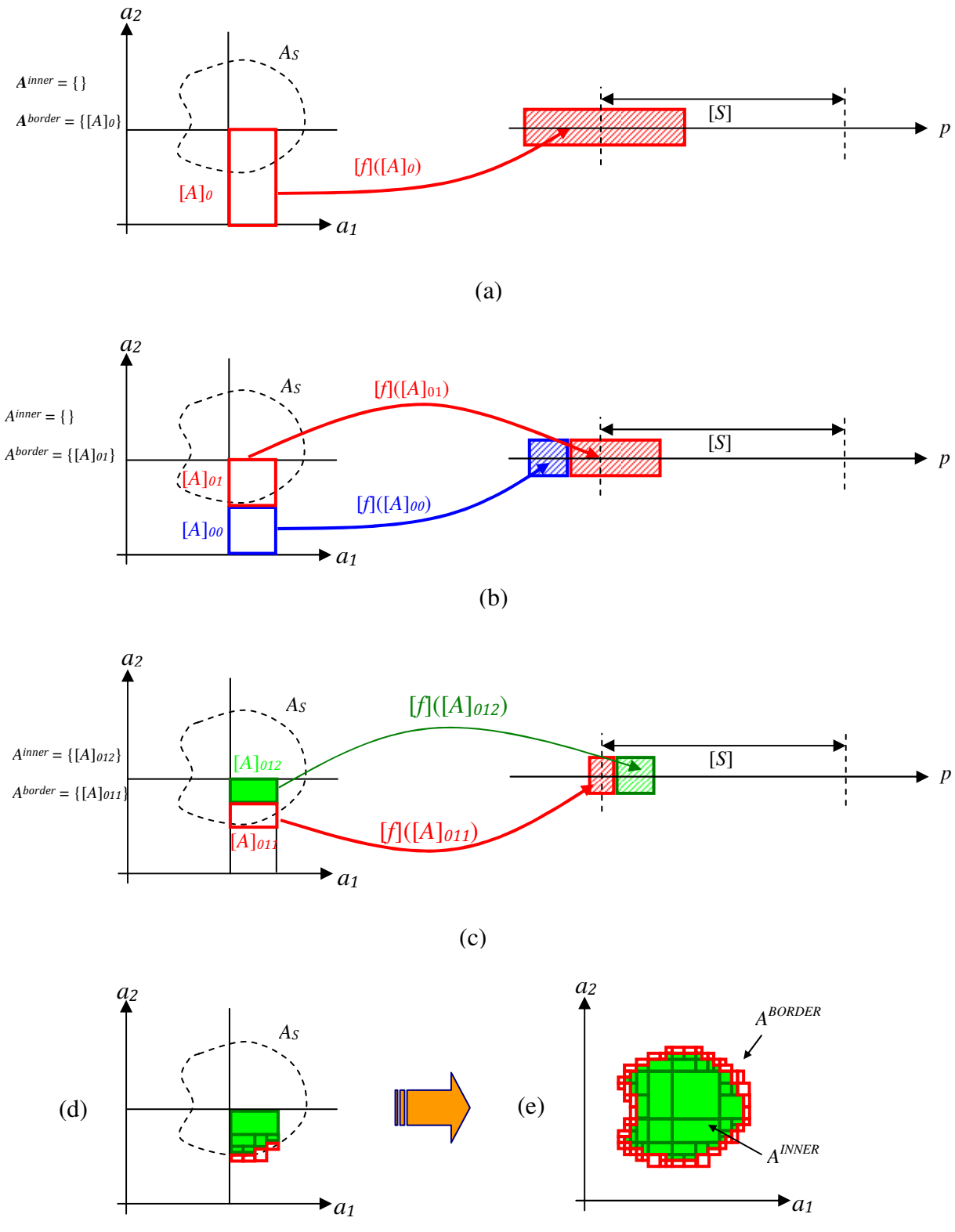


Figure 5-5 Processus de recherche de l'espace solution en pavage par division itérative

5.4.3 Piège de la représentation à deux dimensions d'un ensemble par sous-pavage

L'exemple qui suit met en évidence toute la subtilité de la représentation d'un espace par une réunion de sous-pavés. Nous ne représenterons ici que les pavés de la classe A^{INNER}, mais les mêmes remarques vaudront pour les autres sous espaces.
Considérons que le sous-espace recherché est décrit par une sphère dont l'expression analytique est donnée dans un espace d'ajustement à trois paramètres $A = \{a_1, a_2, a_3\}$ par :

$$K = (a_1 - 3)^2 + (a_2 - 3)^2 + (a_3 - 3)^2$$

Avec : $a_i \in [0,10]$ et K le carré du rayon de la sphère.

En considérant l'expression des spécifications par **S** = $\{1 \leq K \leq 2\}$, nous sommes alors bien en présence de la représentation d'une sphère (boule creuse). Le sous-espace solution est représenté sur la figure 5-6a dans le plan (a_1,a_2) lorsque le paramètre a_3 est limité à $a_3 \in [0,10]$ par une superposition de l'ensemble des sous-pavés intérieurs donnant une représentation d'un sous-espace plein. Si l'on restreint, sur la figure 5-6b, la représentation des pavés uniquement à l'intervalle $a_3 \in [2.98,3.02]$, une zone vide de pavés apparaît alors au centre de la sphère. Cette représentation peut être interprétée comme une « tranche » de cette sphère. Une représentation en deux dimensions peut donc être vue comme un empilement de pavés pouvant cacher des zones vides de solution, ce type de représentation doit donc être interprété avec précaution.

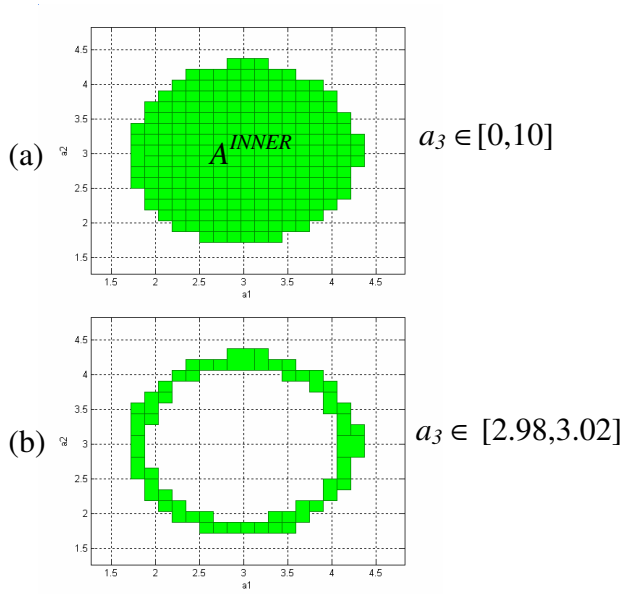


Figure 5-6 Représentation du sous-espace solution dans le plan (a_1,a_2).

Nous venons d'illustrer le concept d'inversion ensembliste. Ce concept nous permet d'obtenir une description explicite de A_S par un pavage en appliquant la procédure de division itérative et un test de classification des pavés. Nous présentons maintenant une approche complémentaire qui propose de limiter la taille du pavage afin de réduire la quantité de mémoire et le temps de calculs qui alourdissent l'utilisation du concept d'inversion ensembliste.

5.5 La réduction de pavé par les « contracteurs »

Nous avons introduit la fonction d'inclusion naturelle et avons montré son utilité et son importance dans le processus d'inversion ensembliste. Néanmoins elle peut introduire un « pessimisme » certain dans l'évaluation du pavé image par rapport à l'ensemble image exact. Ce

pessimisme a pour conséquence de réduire l'efficacité du classement des pavés A^{INNER} ou $A^{OUTSIDE}$ ce qui conduit à classer ces pavés dans les A^{BORDER} (aussi appelé Undetermined). Or les pavés de cette dernière liste devront être divisés afin de d'accroître la résolution de l'approximation selon la propriété de convergence des fonctions d'inclusions [Jaulin 01a, p28]. Or la division dans un espace de dimension élevée (nombre de variables libres > 10) conduit à une explosion du nombre de pavés à traiter. La division, dans le cas de système à grand degré de liberté, doit donc être limitée aux cas de blocages.
Diverses réponses existent pour minimiser l'impact de ce pessimisme. Pour une fonction scalaire élémentaire, il existe plusieurs types (ou définitions) de fonctions d'inclusions associées retournant des bords plus ou moins proche de la solution exacte [*cf.* §5.2]. Contrairement à la version « naturelle », les autres types nécessitent plus de calculs minorant donc l'efficacité du traitement. Avec des pavés de grandes tailles comme c'est le cas au début d'un dimensionnement de circuits, [Jau-01b] a montré que les fonctions d'inclusions naturelles sont au final les moins pessimistes de toutes. Les autres types apporteront une meilleure précision lorsque la taille des pavés sera plus petite, c'est à dire lorsque l'estimation de l'espace faisable (ou plus généralement l'espace solution) sera suffisamment détaillée pour une première approche de conception.
Une alternative au changement de type de fonction d'inclusion pour réduire la taille des pavés avant d'opérer les tests de classification est le recourt aux opérateurs de « réduction » (narrowing operators) encore appelés les opérateurs de « contraction » (contractors).

Les « contracteurs » tel que nous allons les introduire ont été originellement développés pour contribuer à résoudre les problèmes d'optimisation globale. La résolution d'un tel problème passe par la résolution des contraintes générant ainsi l'espace de points faisables dans lequel l'optimisation devra rechercher le point optimal, c'est ce que l'on appelle un CSP (Constraint Satisfaction Problem) [*cf.* §4.1]. En fait, on remarquera qu'un CSP n'est rien d'autre qu'un système d'équations à satisfaire. La même problématique se retrouve donc dans la résolution de systèmes d'équations ou d'inéquations. Cette remarque a conduit certains travaux [Han-03] à introduire la notion de « consistency » indépendamment d'un problème d'optimisation.

Historiquement la résolution du CSP en analyse numérique fut introduite par Waltz [Wal-75]. L'adaptation au calcul par intervalle fut proposée indépendamment par [Dav-87, Cle-87] qui fut nommée ICP (Interval Constraint Propagation). Dans les problèmes d'optimisation, où cette technique est très utilisée, on parle de contraintes ce qui justifie l'emploie du terme « constraint propagation », mais pour ne pas limiter la portée de ce qui suit, nous parlerons de Condition à satisfaire plutôt que de Contraintes.

5.5.1 Introduction des contracteurs dans la résolution d'un problème de CSP

Les principes mathématiques de la résolution d'un CSP vont être introduits sur une condition d'égalité. Or les cas de dimensionnement et par suite les cas d'optimisations feront apparaître des conditions d'appartenance et des conditions d'inégalités. Nous commencerons donc par montrer que ces cas peuvent ce ramener à une condition d'égalité rendant alors notre analyse plus générale.

Un problème de satisfaction des contraintes (CSP) [*cf.* §4.1] peut se résumer à une résolution du système d'équations suivant :

$f(a) = 0$, équ. A
$a \in [A]$.

Dans ce système f et une fonction définit par f :$\mathbf{R}^n \rightarrow \mathbf{R}$. L'espace de recherche des solutions est contraint au pavé $[A]$ devenant le domaine de f. La résolution du système (soit aussi le CSP) revient à trouver l'ensemble $A_S = \{ a \in [A] / f(a) = 0 \}$ des points qui vérifient les deux conditions (soient $f(a) = 0$ et $a \in [A]$). En optimisation l'ensemble A_S s'appelle l'ensemble « faisable ». La première condition restreint l'espace solution à l'ensemble $A_{Sf} = \{ a \in \mathbf{R}^n / f(a) = 0 \}$. Le respect simultané des deux conditions conduit à l'ensemble $A_S = A_{Sf} \cap [A]$ correspondant clairement à sa définition : $A_S = \{ a \in [A] / f(a) = 0 \}$.

5.5.1.1 Condition d'appartenance

Dans le cas d'un dimensionnement, nous exprimons la condition qu'une performance doit satisfaire par : $p \in [S]$, où p est une valeur prise par une performance et $[S]$ est l'intervalle spécifiant les valeurs acceptables de celle-ci. L'objectif à satisfaire est alors de trouver l'ensemble des valeurs des variables ajustements a qui vont satisfaire le système suivant :

$p = f(a) \in [S]$, équ. B
$a \in [A]$.

L'ensemble des solutions de ce système devra vérifier la définition suivante :

$A_S = A_{Sf} \cap [A] = \{ a \in [A] / f(a) \in [S] \}$ avec $A_{Sf} = \{ a \in \mathbf{R}^n / f(a) \in [S] \}$.

En transposant ce problème en calcul par intervalle, on obtient le système suivant :

$[P] = [f]([A]) = [S]$, équ. C
$a \in [A]$.

Ce système pourra être transformé en un système équivalent mais conforme à un CSP simplifié:

$[f]([A]) - [S] = [0,0] = 0$, équ. D
$a \in [A]$.

Par le théorème fondamentale de Moore, on vérifie la relation suivante : $f([A]) \subseteq [f]([A])$. Ainsi l'ensemble image vérifiera : $P_S = \{ p \in \mathbf{P} / \exists a \in [A] \wedge p = f(a) \in [S] \}$. Par suite l'ensemble solution du système vérifiera : $A_S = f^{-1}([S]) = \{ a \in [A] / f(a) \in [S] \}$

5.5.1.2 Condition d'inégalité

Les conditions d'inégalités peuvent se transformer par l'introduction de variables auxiliaires (*slack variables*) [Han-03, ch6]. Nous supposerons ici que c'est toujours possible, sinon on se reportera à [Han-03, ch13] pour des transformations plus subtiles. Illustrons cette affirmation par un exemple simple. Soit l'inégalité suivante $[a][x] + [b] \leq 0$, en introduisant la variable auxiliaire [c] =]-∞,0], l'inégalité impose que le membre de gauche appartienne à l'intervalle [c] ce qui donne l'équivalence suivante :

$[a][x] + [b] \leq 0$ ™ $[a][x] + [b] = [c] =]-\infty,0]$.

La dernière expression peut donc être transformée en : $[a][x] + [b] - [c] = [0,0] = 0$, qui est une condition d'égalité.

Avec ces deux transformations, on pourra transformer un CSP général (référence équation générale CSP) en un CSP simplifiée (équ. A).

5.5.1.3 Notion de « *consistency* »

Une autre notion importante pour comprendre la suite, est la notion de pavé consistant avec une condition. Ce point est particulièrement bien abordé dans [Han-03, ch10] bien qu'il a eu une approche différente de celle des créateurs de l'ICP que nous aborderons plus loin.
Considérons une équation implicite $f(x,y) = 0$ et admettons que les variables x et y appartiennent respectivement aux intervalles $[X] \subset \mathbf{R}$ et $[Y] \subset \mathbf{R}$. Nous pouvons dire qu'un point $(x,y) \in [X]\times[Y]$ est consistant avec la condition d'égalité $f(x,y) = 0$ si :

$$(\forall x \in [X], \exists y \in [Y] / f(x,y) = 0) \wedge (\forall y \in [Y], \exists x \in [X] / f(x,y) = 0).$$

Supposons qu'il existe un sous ensemble U de valeurs de $x \in [X]$ pour lesquels il n'existe aucune valeur $y \in [Y]$ vérifiant la condition, alors ce sous ensemble U peut être exclu lors de la recherche des points solutions de la condition $f(x,y) = 0$ pour inconsistance avec la condition.

Supposons maintenant que nous recherchions le pavé contenant les points solutions de la condition d'égalité $f(x,y) = 0$, alors le sous pavé $[U]\times[Y]$ est un pavé que l'on rejette immédiatement car inconsistant avec la condition. Le concept de consistance ou d'inconsistance est fondamental pour les opérateurs de réduction de pavé sans perte de point solution. Il a conduit à deux algorithmes fondateurs : « *Box consistency* » et « *Hull consistency* » dont est inspiré l'algorithme de « *Forward-Backward* » que nous allons présenter ci-dessous.

5.5.1.4 Notion de « Contracteur Extérieur »

Pour résoudre efficacement un problème, il faut parfois construire un second problème plus complexe mais possédant de bonnes propriétés. Une partie des bonnes propriétés permettent de le résoudre efficacement. Une autre bonne propriété est que les solutions du second sont aussi les solutions du premier. Or [Jau-01b] introduit pour cela la notion de « Contracteur Extérieur ».

Soient deux CSP : H_1 et H_2. Si l'on vérifie :

- toutes les variables de H_1 sont aussi des variables de H_2,
- si un point x est solution de H_1 alors il est aussi solution de H_2;

alors on dira que :

- H_2 découle de H_1,
- H_2 est consistant avec H_1,
- H_2 est une approximation « extérieure » de H_1 car $dim(H_2) > dim(H_1)$,
- $H_1 \Rightarrow H_2$, tous les points solutions de H_1 sont aussi solutions de H_2.

Le CSP H_2 étant plus vaste que H_1, il comporte plus de variables. Ainsi on peut écrire $a_2 = (a_1,u)^T$ où a_1 est le vecteur vérifiant H_2 et H_1 et u le vecteur additionnel ne vérifiant que H_2.
Compte tenu des bonnes propriétés attendues pour H_2, appliquer la contraction à H_1 se fera en trois étapes :

trouver le CSP « externe » H_2 pour lequel il existe des opérateurs de contraction efficaces,

contracter le domaine de recherche satisfaisant le CSP H_2,
reporter le sous domaine contracté de H_2 dans le domaine de H_1.

Pour illustrer ces notions, prenons l'exemple d'un problème contraint *H*.

$$H:\begin{cases} f(a_1,a_2): {a_1}^2+{a_2}^2=2a_1 \\ a_1\in[-1,+1] \\ a_2\in[-1,+1] \end{cases}$$

Le problème *H* est mal « conditionné » avec la présence de multi occurrence de la variable a_1. En introduisant les variables auxiliaires *r* et θ, on peut aussi trouver un CSP « externe » H_1 :

$$H_1:\begin{cases} r^2\cos^2\theta+r^2\sin^2\theta=2r\cos\theta \\ a_1=r\cos\theta \\ a_2=r\sin\theta \\ a_1\in[-1,+1] \\ a_2\in[-1,+1] \\ r\in[0,+\infty] \\ \theta\in[0,2\pi] \end{cases} \Leftrightarrow H_1:\begin{cases} r=2\cos\theta \\ a_1=r\cos\theta \\ a_2=r\sin\theta \\ a_1\in[-1,+1] \\ a_2\in[-1,+1] \\ r\in[0,+\infty] \\ \theta\in[0,2\pi] \end{cases}$$

En introduisant un développement limité au premier ordre au voisinage d'un point $(a_1,a_2) = (\alpha,\alpha)$ sur la fonction $f(a_1,a_2)$:

$$f(a_1,a_2)_\alpha \simeq f(\alpha,\alpha)+(a_1-\alpha)\frac{\partial}{\partial a_1}\left(f'(\alpha,\alpha)\right)+(a_2-\alpha)\frac{\partial}{\partial a_2}\left(f'(\alpha,\alpha)\right),$$

le problème *H* peut également s'exprimer en un autre problème H_2:

$$H_2:\begin{cases} 4\alpha^2+2(u_1-\alpha)+2(u_2-\alpha)=2\alpha \\ \alpha\in[-1,+1] \\ u_1\in[\alpha-\varepsilon,\alpha+\varepsilon] \\ u_2\in[\alpha-\varepsilon,\alpha+\varepsilon] \end{cases} \Leftrightarrow H_2:\begin{cases} 2\alpha^2-3\alpha+a_1+a_2=0 \\ \alpha\in[-1,+1] \\ u_1\in[\alpha-\varepsilon,\alpha+\varepsilon] \\ u_2\in[\alpha-\varepsilon,\alpha+\varepsilon] \end{cases}$$

Où ε fixe la précision de l'approximation.

5.5.1.5 Existence de contracteurs efficaces

Pour démontrer l'existence d'un contracteur efficace (efficace dans le sens où cet opérateur réduit un pavé sans utiliser la division ou bissection), il faut introduire la notion de « finite subsolver » [Jau-01b, p67-§4.2.1, et p70-§4.1]. Cette notion est une restriction du « contracteur extérieur ». Cette fois on cherche un « mini contracteur » qui peut opérer sur quelques composants seulement.

Soit toujours le CSP : $f(a) = 0$ avec $a \in [A]$. On décompose le vecteur a en divers sous vecteurs par $a = (a_a, a_i, a_j, a_b)^T$ où a_a, a_i, a_j et a_b sont indépendants (au sens des vecteurs d'une base vectorielle). Un « mini contracteur » est alors un algorithme (un opérateur) $\Phi : a_i \rightarrow \Phi(a_i)$ tel que l'on vérifie :

$f(a) = 0 \Rightarrow a_j \in \Phi(a_i)$.

S'il existe une fonction d'inclusion $[\Phi]$ pour l'opérateur Φ alors selon les propriétés des fonctions d'inclusions, on aura $\Phi([a_i]) \subset [\Phi]([a_i])$.

Le théorème [Jau-01b, p70-§4.1] conduit alors à caractériser l'opérateur de contraction C pour la condition $f(a)=0$ par la relation :

$$C([a]) = ([a_a], [a_i], [a_j] \cap [\Phi]([a_i]), [a_b])^T \subset ([a_a], [a_i], [a_j], [a_b])^T = [a].$$

5.5.1.6 Exemple de mise en application de cette notion

Soit la condition à vérifier : $a_3 = a_1 * a_2$, il existe 3 « mini contracteurs » Φ_1, Φ_2, Φ_3 définis par:

$\Phi_1(a_2, a_3)$ et $a_1 \in \Phi_1(a_2, a_3)$;
$\Phi_2(a_1, a_3)$ et $a_2 \in \Phi_2(a_1, a_3)$;
$\Phi_3(a_1, a_3)$ et $a_2 \in \Phi_3(a_1, a_3)$.

En fin de procédure de contraction,
$[a_1]$ sera remplacé par $[a_1] \cap [a_3]/[a_2]$;
$[a_2]$ sera remplacé par $[a_2] \cap [a_3]/[a_1]$;
$[a_3]$ sera remplacé par $[a_3] \cap [a_1]*[a_2]$.

5.5.2 Première application des contracteurs dans un cas de dimensionnement

On considère une fonction $f : \mathbf{R}^n \rightarrow \mathbf{R}$, $a \rightarrow p$, devant satisfaire la condition : $p = f(a) \in [S]$ avec $a = (a_1,..., a_n)^T$. L'ensemble solution de cette condition est :

$$A_{Sf} = \{ a \in \mathbf{R}^n / f(a) \in [S] \} = f^{-1}([S]).$$

Plutôt que de commencer la recherche dans tout l'espace $\mathbf{R}^n$, on décide de restreindre l'espace de recherche à un pavé initial $[A]_o$. L'ensemble solution restreint sera donc $A_S = A_{Sf} \cap [A]_o$ du domaine $\mathbf{R}^n$.

Cependant, malgré la limitation du domaine de recherche, il est possible que tous les points du pavé initial $[A]_o$ ne soient pas des points solutions. On souhaite donc réduire ce pavé $[A]_o$ sous la condition $f(a) \in [S]$ pour extraire le pavé réduit $[A]_1$ tel que $A_S \subsetneq [A]_1 \subsetneq [A]_o$. Les objectifs annexes visés sont :

la réduction doit se faire sans bissection,
la réduction doit contenir avec garantie l'ensemble solution $A_S = A_{Sf} \cap [A]_o$.

Il vient alors $A_S = A_{Sf} \cap [A]_o \subset [A_1] \subset [A]_o$. En calculant l'intersection avec A_{Sf} pour chaque membre de la relation précédente on obtient :

$$\begin{array}{lllll} & A_{Sf} \cap (A_{Sf} \cap [A]_o) & \subseteq & A_{Sf} \cap [A]_1 & \subseteq & A_{Sf} \cap [A]_o, \\ \Rightarrow & A_{Sf} \cap [A]_o & \subseteq & A_{Sf} \cap [A]_1 & \subseteq & A_{Sf} \cap [A]_o, \\ \Rightarrow & A_{Sf} \cap [A]_o & = & A_{Sf} \cap [A]_1 & & \end{array}$$

Soit encore $A_S = A_{Sf} \cap [A]_0 = A_{Sf} \cap [A]_1$ qui prouve que l'on ne perd aucun point solution en passant du pavé $[A]_0$ au pavé réduit $[A]_1$.

La démarche précédente peut être généralisée à d'autres problèmes en reformulant les objectifs sans modifier la démarche. Partant d'un pavé domaine initial quelconque $[A]_0$, on veut trouver un nouveau pavé $[A]_1$ contenant avec garantie A_S. Il vient alors que $A_S \subset [A]_1 \subset [A]_0$. Un opérateur qui transforme un pavé $[A]_0$ en un pavé $[A]_1$ vérifiant les inclusions précédentes sera appelé un « contracteur » l'on parle aussi d'opérateur de réduction (*Narrowing Operator*). Il réduit donc un pavé sans perdre aucun point solution du problème CSP.

Le processus de réduction peut alors être réitéré plusieurs fois jusqu'à ce que la réduction converge vers le pavé minimal $[A_S^{min}] = [\ A_{Sf} \cap [A]_0\ ]$. Les diverses entités du principe sont illustrées sur la figure 7 ci-dessous :

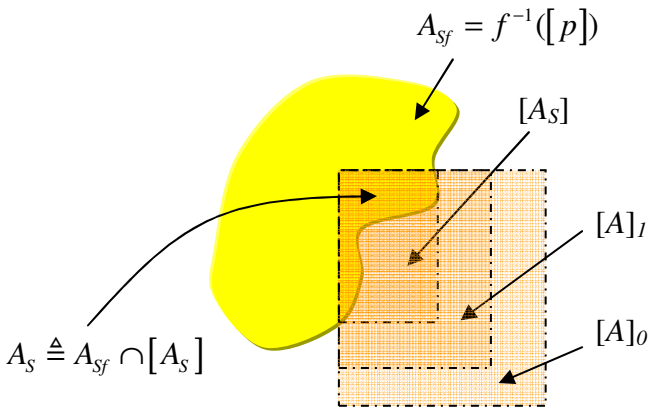


Figure 5-7 Illustration du processus de réduction d'un pavé sous contraintes

Le principe d'application d'un contracteur étant posé, il nous faut maintenant détailler comment construire (ou calculer) le pavé réduit. Pour la simplicité de l'exposé, nous analyserons un cas d'égalité dans le domaine réel. L'adaptation aux calculs par intervalle sera introduite ultérieurement.

5.5.3 Mini-contraction suivant une seule direction de l'espace.

Les techniques de construction des contracteurs s'appuient sur l'utilisation de fonctions «termes». Une fonction «terme» possède la propriété de pouvoir isoler chaque variable a_i dans l'expression de cette fonction $f(a_1,..., a_i,..., a_n) = p$. La mise en œuvre pratique de ces techniques fait appelle à la théorie des graphes avec la notion de fonctions acycliques. Ces deux notions sont parfaitement équivalentes permettant de passer de l'une à l'autre sans équivoque.

Si on peut isoler la composante a_i dans l'expression de la fonction « terme » $f(a) = p$, cela revient à dire qu'il existe une fonction g_i qui satisfasse l'équivalence suivante :

$$f(a) = p \text{ ™ } a_i = g_i({}^i a, p) \text{ où } {}^i a = (a_1,..., a_{i-1}, a_{i+1},..., a_n)^T .$$

La fonction g_i est alors appelée « fonction associée à a_i ». On remarquera que ces définitions sont équivalentes à celles utilisées dans les études des fonctions implicites.

5.5.3.1 Théorème de la Projection de la fonction associée

Soit une fonction terme $f(a)$ vérifiant la condition $f(a) = p$ avec $p \in [S]$ pour laquelle il existe la fonction associée g_i pour la composante a_i avec sa fonction d'inclusion $[g_i]$, soit l'opérateur de projection $proj_i$ de p sur l'axe a_i (suivant la i^{ième} direction), alors nous avons :

$$proj_i\,(\mathbf{A_{Sf}} \cap [A]_o) \subset [g_i]([\,^i a],[S]) \cap [a_i] \text{ où } [a_i] \text{ est la } i^{ième} \text{ composante du pavé } [A]_o.$$

De plus si $[g_i]$ est minimale et g_i continue alors l'inclusion devient une égalité.

5.5.3.2 Démonstration

$$\begin{aligned} & f(a_1,\dots,a_n) \in [p] && \Leftrightarrow && (\exists p \in [p] \,/\, f(a) = p) \\ & && \Leftrightarrow && (\exists p \in [p] \,/\, a_i = g_i(\,^i a, p)) \end{aligned}$$

$$\begin{aligned} & proj_i\,(A_S \cap [A]_o) = \{\, a_i \in [a_i] \,/\, (\forall k \neq i, \exists\, a_k \in [a_k]) \text{ tel que } f(a) \in p \,\} \\ \Leftrightarrow\quad & proj_i\,(A_S \cap [A]_o) = \{\, a_i \in [a_i] \,/\, (\forall k \neq i, \exists\, a_k \in [a_k]) \wedge (p \in [p] \,/\, a_i = g_i(\,^i a, p)) \,\} \\ \Leftrightarrow\quad & proj_i\,(A_S \cap [A]_o) = \{\, a_i \in [a_i] \,/\, a_i \in g_i([\,^i a],[p]) \,\} \end{aligned}$$

En introduisant sa fonction d'inclusion $[g_i]$ et en s'appuyant sur la propriété des fonctions d'inclusion [Moo-79], on aura aussi :

$$\begin{aligned} & g_i([\,^i a],[S]) \subset [g_i]([\,^i a],[S]) \\ \Leftrightarrow\quad & proj_i\,(A_{Sf} \cap [A]_o) \subset [g_i]([\,^i a],[S]) \cap [a_i] \\ \Leftrightarrow\quad & proj_i\,(A_{Sf} \cap [A]_o) \subsetneq [\, proj_i\,(A_{Sf} \cap [A_o])\,] \equiv\!\!\!\cdot\ [\, proj_i\,](A_{Sf} \cap [A]_o). \end{aligned}$$

Au final on pourra écrire :

$$proj_i\,(A_S) \subsetneq [\, proj_i\,(A_S)\,] \equiv\!\!\!\cdot\ [\, proj_i\,](A_S),$$

où l'inclusion deviendra une égalité si g_i est continue et $[g_i]$ est minimale.

5.5.3.3 Utilisation des fonctions acycliques pour construire les contracteurs

Les fonctions acycliques sont issues de la théorie des graphes et permettent de décomposer une expression analytique en une série d'opérations élémentaires qu'un algorithme pourra traiter. Ces fonctions acycliques sont équivalentes aux fonctions « termes » précédentes comme le montre Jaulin dans [Jau-01c, p 43-§3.5.2].

5.5.3.4 Définition d'une fonction acyclique

La fonction $f(a)$ est acyclique si elle peut être représentée par un arbre dans lequel :

- chaque feuille est soit une composante a_i de a soit un réel quelconque,
- chaque nœud n qui n'est pas une feuille est associé soit à un opérateur arithmétique (opérateur binaire) {+, *}, soit a une fonction élémentaire (opérateur unaire) {*sin*, *cos*, *exp*, *ln*, *x*, x^2, *inv*(*x*), …},
- chaque variable a_i doit apparaître au plus une fois dans l'arbre.

Pour la compréhension de cette définition, nous rappelons ci-dessous quelques points importants de la théorie des graphes.

- Un arbre est un graphe connexe sans cycle. On parle d'arbre enraciné, c'est-à-dire qu'un des nœuds est désigné comme racine. C'est exactement ce que l'on utilise dans la description d'un circuit électrique lors de la création de la netlist de SPICE où le nœud 0 est arbitrairement la référence des potentiels.
- Les opérateurs {-, /} étant les opérateurs inverses des deux opérateurs arithmétiques retenus, ils ne doivent pas apparaître dans les expressions de la fonction *f*. On transformera les expressions suivantes :
  $(a_1 - a_2) \Leftrightarrow (a_1 + (-1)*a_2)$
  et $a_1/a_2 \Leftrightarrow (a_1 * inv(a_2))$.
- Restreindre l'analyse des contracteurs aux fonctions acycliques n'est pas vraiment une limitation, car en général, les expressions que nous utilisons peuvent par l'introduction de variables intermédiaires (*slack variables*) être transformées en fonctions acycliques comme le montre l'exemple ci-dessous.

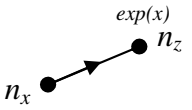
Exemple :

Soit la fonction à décomposer $f(a) = a_1 - exp(a_1.a_2) \in [S]$ avec $a = (a_1,a_2)^T$. En introduisant la variable auxiliaire $u \in [-\infty,+\infty]$ on obtient un système de deux équations:

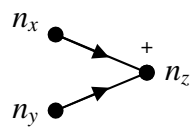
$$f_1(a,u) = a_1 - u \in [0,0]$$
$$f_2(a,u) = u - exp(a_1.a_2) \in [S].$$

Ainsi chaque variable n'apparaît qu'une fois dans chaque expression et l'on dispose maintenant de deux fonctions acycliques. On abordera la résolution d'un système de deux fonctions plus loin. Une fois que la fonction est traduite en graphe, une procédure d'évaluation directe va propager les données numériques des feuilles vers la racine. Ensuite une seconde étape va rétro propager, de la racine vers les feuilles, les valeurs obtenues au cours de la phase directe. L'ensemble des deux étapes (directe + arrière) permet de « propager » les données sur l'arbre. Cette procédure en deux étapes fut introduite par [Ben-99] avant de le transposer ensuite au calcul par intervalle.

On progresse dans l'arbre en passant d'un nœud à un ou deux nœuds selon que l'opération est unaire ou binaire. A chaque nœud on associe une variable et son domaine. Analysons les deux cas de figures qui peuvent se présenter :



- pour un opérateur unaire comme $z = exp(x)$, le nœud n_z est lié par un graphe au nœud n_x. Le graphe porte l'operateur unaire, ici *exp*().
  En phase directe le domaine de z, D(z) est modifié par le domaine de x, D(x) via la relation D(z) = *exp*(D(x)).
  En phase arrière le domaine D(x) est modifié par D(x) = f^{-1}(D(z)) ∩ D(x) = *ln*(D(z)) ∩ D(x) en application du théorème (Th Pfa).



- pour un opérateur binaire comme $z = x + y$, le nœud n_z reçoit deux graphes venant respectivement des noeuds n_x et n_z.
  En phase directe on aura D(z) = D(x) + D(y).
  En phase arrières on aura respectivement : D(x) = (D(z) – D(y)) ∩ D(x) et D(y) = (D(z) – D(x))) ∩ D(y) en application du théorème (Th Pfa).

On retrouve dans ces deux cas de figures, l'introduction des opérations inverses propre à chaque relation élémentaire.

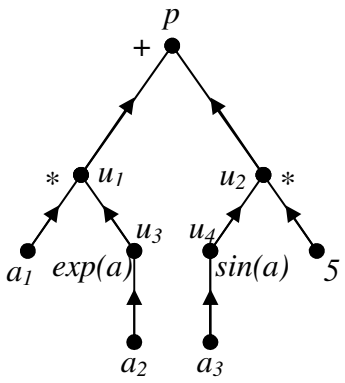
Remarque : l'optimalité de la réduction est une conséquence directe du fait qu'un arbre n'a pas de cycle [Hyv-92].
Remarque : résoudre UNE condition du type $f(a) \in p$ conduit donc à construire un nouveau système à K conditions primitives permettant de construire l'arbre acyclique. On est ici dans l'illustration de l'importance de la notion de « contracteur externe ».

Exemple de décomposition en série d'opérations inversibles :

Soit la fonction $p = f(a_1, a_2, a_3) = a_1 * exp(a_2) + 5 * \sin(a_3)$, la décomposition suivant un arbre acyclique de la fonction revient à décomposer la fonction condition d'un CSP en un système de conditions primitives. Cette décomposition peut être vue comme un CSP « externe » H_2 comparativement à la fonction originelle qui forme le CSP H_1.

La décomposé en arbre acyclique de la fonction f nécessite l'introduction des variables auxiliaires u_1, u_2, u_3, u_4, initialisées à $[-\infty,+\infty]$:

$p = u_1 + u_2$; équ. 1
$u_1 = a_1 * u_3$; équ. 2
$u_3 = exp(a_2)$; équ. 3
$u_2 = 5 * u_4$; équ. 4
$u_4 = sin(a_3)$; équ. 5



La décomposition en arbre acyclique revient à décomposer la condition en un ensemble de conditions primitives conformément à la définition des fonctions acycliques.
Cette décomposition peut être vue comme un nouveau système de conditions correspondant à un CSP « externe » H_2 comparativement à un CSP H_1.

Remarque : l'analyse du système d'équations du CSP H_2 montre que :
La performance p n'apparaît qu'une seule fois et à gauche ;
les a_i n'apparaissent qu'une fois dans le système d'équations et à droite ;
les u_i n'apparaissent respectivement une fois à gauche et une seule fois à droite.

Comme chaque opérateur primitif est inversible on peut réécrire le CSP H_2 en un nouveau CSP externe à H_2 que nous appellerons H_3 par :

$u_2 = p - u_1$;
$u_1 = a_1 * u_3$;
$u_3 = exp(a_2)$;
$u_2 = 5 * u_4$;
$a_3 = sin^{-1}(u_4)$;

ainsi on peut déterminer le domaine de a_3 par :

$D(a_3) = [a_3]_0 \cap [sin^{-1}](1/5 * ([S] - [a_1]_0 * [exp]([a_2]_0)))$;
$\Leftrightarrow$ $D(a_3) = [a_3]_0 \cap [g_3]([(a_1, a_2)]_0, [S])$;
$\Leftrightarrow$ $D(a_3) = [a_3]_0 \cap [g_3]([^3a]_0, [S])$;

$\Leftrightarrow \quad D(a_3) = [a_3]_0 \cap [proj_3](A_{Sf} \cap [A]_0)$;

La résolution numérique du CSP H_3 est faite selon l'algorithme «*Forward-Backward* Propagation » (*FwBw*) qui s'énonce :

> « Projection directe » *For k* =1 to 5 (équation 1 à 5)
> Projeter les membres de droite vers les membres de gauches ;
> *End*
> « Projection arrière » *For k* = 5 to 1
> Projeter les membres de gauches vers les membres de droite;

On constate qu'au cours des différentes étapes de cet algorithme, les variables (d'ajustements *a* et auxiliaires *u*) sont propagées au travers de toutes les équations du système de H_3 ; ceci justifie l'appellation « propagation des conditions ».

Remarque : une itération de l'algorithme FwBw, ne réduit pas forcement le pavé initial $[A]_0$ car les variables auxiliaires *u* ont été initialisées à $]-\infty, +\infty[$. Dans la pratique, il faut souvent deux à trois itérations pour obtenir une contraction du pavé initial.

En plaçant l'algorithme dans une boucle, on peut faire converger le pavé initial vers l'un ou l'autre des trois cas suivants :

- soit un pavé vide (pavé dont une dimension est nulle) si aucun point solution n'appartient à ce pavé,
- soit un pavé bloquant sur un « point fixe » (*cf.* §5.5.6),
- soit LE pavé minimum $[A_S^{min}] = [A_{Sf} \cap [A]_0]$

5.5.4 Introduction des « P-opérateurs »

Nous allons introduire maintenant des opérateurs contractants qui pourront remplacer directement les fonctions élémentaires. Ces opérateurs découlent de l'algorithme FwBw mais rejettent les prétraitements nécessaires des équations. En ce sens ils n'auront pas la même efficacité que l'application de l'algorithme FwBw, néanmoins sur les quelques exemples concrets où nous avons travaillés, les temps de calculs restent du même ordre de grandeurs.

Considérons l'expression de la fonction $p = a_1.a_2^2 + a_2.a_3$ exprimant les performances à partir d'un espace d'ajustements à trois paramètres a_1, a_2 et a_3. Cette expression est une fonction terme qui peut être décomposé en sous fonctions élémentaires selon le principe des fonctions acycliques. Pour chaque fonction élémentaire, la projection suivant une direction devient un mini contracteur efficace.

Le tableau 1 détaille le processus de décomposition de la fonction en quatre primitives. Une itération correspond donc ici à l'exécution de ces quatre primitives. Ces primitives sont converties en contracteurs que nous appellerons les « P-opérateurs ». Le détail de ces opérateurs est donné dans le tableau 5-1.

Le processus itératif sera reconduit jusqu'à ce qu'aucune contraction ne soit observée, c'est à dire qu'aucune variable n'a plus était réduite.

propagation

Etapes	Primitives	Expressions contractées
1	$[u_1] := [u_1] \cap ([a_2]*[a_2])$	$u_1 = a_2^2$
	$[a_2] := [a_2] \cap \text{sqrt}[u_1]$	
2	$[u_2] := [u_2] \cap ([a_1]*[u_1])$	$u_2 = a_1*a_2^2$
	$[a_1] := [a_1] \cap ([u_2]/[u_1])$	
	$[u_1] := [u_1] \cap ([u_2]/[a_1])$	
3	$[u_3] := [u_3] \cap ([a_2]*[a_3])$	$u_3 = a_2*a_3$
	$[a_3] := [a_3] \cap ([u_3]/[a_2])$	
	$[a_2] := [a_2] \cap ([u_3]/[a_3])$	
4	$[p] := [p] \cap ([u_2] + [u_3])$	$p = u_2 + u_3$
	$[u_2] := [u_2] \cap ([p] - [u_3])$	
	$[u_3] := [u_3] \cap ([p] - [u_2])$	

Tableau 5-4 Séquence d'exécution des primitives de la fonction $p = a_1.a_2^2 + a_2.a_3$

A l'étape 1, la primitive $u_1 = a_2^2$ est contractée à l'aide du P-opérateur unaire *psquare* où deux opérations sont nécessaires en faisant intervenir uniquement les deux variables u_1 et a_2. La première opération correspond à la phase directe (*forward*) et la seconde correspond à la phase arrière (*backward*). L'opérateur binaire *pmult* est utilisé pour les étapes 2 et 3 et l'opérateur binaire *pplus* pour l'étape 4. L'utilisation de trois paramètres pour ce type d'opérateur nécessite donc par conséquent trois opérations par étape.

psquare($[z_k]$, $[x_k]$)	$[z_{k+1}] := [z_k] \cap ([x_k]*[x_k])$ $[x_{k+1}] := [x_k] \cap \text{squart}[z_{k+1}]$
pmult($[z_k]$, $[x_k]$, $[y_k]$)	$[z_{k+1}] := [z_k] \cap ([x_k]*[y_k])$ $[y_{k+1}] := [y_k] \cap ([z_{k+1}]/[x_k])$ $[x_{k+1}] := [x_k] \cap ([z_{k+1}]/[y_{k+1}])$
pplus($[z_k]$, $[x_k]$, $[y_k]$)	$[z_{k+1}] := [z_k] \cap ([x_k]+[y_k])$ $[y_{k+1}] := [y_k] \cap ([z_{k+1}]-[x_k])$ $[x_{k+1}] := [x_k] \cap ([z_{k+1}]-[y_{k+1}])$

Tableau 5-5 Opérateurs de contraction par propagation (p-opérateurs)

Cet exemple met en pratique une technique particulière de contraction inspiré de l'algorithme « *forward-backward* » et profite des mêmes avantages. Notamment, ce sont les contracteurs les plus efficaces lorsque les pavés sont de grandes tailles [Jau-01b, p91]. D'autre part ils ne nécessitent pas de pré traitement des équations et la transposition est quasi systématique voir même automatisable (voir le programme Quimper). C'est l'ensemble de ces points qui nous a poussé à retenir ce type de contracteurs.

5.5.5 Généralisation à un CSP comportant plusieurs conditions

Soit le CSP H défini par :

$f_1(a) = 0$
$f_2(a) = 0$
$a \in [A]$

La généralisation est triviale du fait que l'on sait transformer un CSP en un CSP « externe ». On traite alors la condition $f_1(a) = 0$ avec $a \in [A]$ comme un CSP H_1, puis on traite la condition $f_2(a) = 0$ avec $a \in [A]$ comme un CSP H_2.

Au final le CSP H doit satisfaire la CSP H_1 & satisfaire le CSP H_2. L'ensemble solution du CSP H_1 est alors $A_S(H_1)$ et l'ensemble solution du CSP H_2 sera $A_S(H_2)$. On aura donc:

$$
\begin{aligned}
& A_S(H) = A_S(H_1) \cap A_S(H_2); \\
\Leftrightarrow \quad & A_S(H) = (A_{Sf}(H_1) \cap [A]) \cap (A_{Sf}(H_2) \cap [A]); \\
\Leftrightarrow \quad & A_S(H) = (A_{Sf}(H_1) \cap A_{Sf}(H_2)) \cap [A]; \\
\Leftrightarrow \quad & A_S(H) = A_{Sf}(H) \cap [A] \equiv \{a \in [A] \,/\, f_1(a) = 0 \wedge f_2(a) = 0\};
\end{aligned}
$$

ce qui prouve que $A_S(H)$ est bien l'espace solution du CSP H.

La décomposition du CSP en sous CSP (H_1 et H_2) permet alors de considérer qu'un CSP aura autant de contracteurs que de conditions. Dans l'exemple précédent, on aura les deux contracteurs suivants :

- le contracteur C_1 pour la condition $f_1(a) = 0$,
- le contracteur C_2 pour la condition $f_2(a) = 0$.

On réduira alors le pavé initial $[A]_0$ avec le contracteur C_1 pour obtenir le pavé réduit $[A]_{0\text{-}1}$. Ce pavé sera ensuite réduit par le second contracteur C_2 pour obtenir au final le pavé réduit $[A]_1$. L'opérateur de contraction global C sera donc défini par :

$$
\begin{aligned}
& C \equiv C_2 \circ C_1, \\
\Leftrightarrow \quad & [A]_1 = C([A]_0) = C_2 \circ C_1([A]_0).
\end{aligned}
$$

Chaque contracteur C_i précédent, fera appelle à une collection de « mini-contracteurs » (c_1, …, c_m). Ces « mini-contracteurs » seront obtenus par la décomposition des fonctions « termes » (fonctions acycliques). Chaque « mini-contracteur » utilise alors la projection suivant une direction de l'espace, projection que l'on peut effectivement calculer par le fait que la décomposition des fonctions acycliques fait apparaître des fonctions élémentaires dont on connaît la forme explicite de la fonction réciproque. L'algorithme FwBw permet alors de propager les résultats de chaque « mini-contracteur » au travers de l'arbre acyclique pour apparaître alors comme une seule opération de contraction pour la condition C_i. On itèrera ce processus au travers de toutes les conditions du CSP pour faire apparaître l'opérateur de contraction global C.

5.5.6 Synthèse des principes des contracteurs

L'outil ICP tel qu'il a été introduit brièvement ci-dessus ouvre la porte vers les traitements de problèmes à grande dimension (plus de 10 variables libres) car aucune bissection n'est nécessaire. Si un problème ne comporte aucune solution, alors la contraction retournera un domaine vide rapidement sans aucune bissection. Ce genre de réponse rapide permet de rejeter efficacement tout un ensemble de pavés et améliorer l'exploration de l'espace de conception en focalisant le temps de calcul sur les pavés délicat qui définissent la frontière de l'espace solution s'il y en a.

Cependant les contracteurs ont une limite. Dans certain cas les contracteurs bloquent sur un « point fixe », c'est ce que Jaulin appelle la « *localy consistency* » [Jau-01b, p30], c'est-à-dire pour tout indice $k > k_0$ on observera un état stationnaire des pavés contractés en vérifiant :

$$C([A]_k) = [A]_k = [A]_{k0}.$$

Pour sortir de ce blocage, plusieurs idées ont été proposées dans [Jau-01b] dont en voici deux que nous avons utilisées :

- l'ajout de contraintes redondantes ;
- la bissection.

Parmi ces deux propositions, la bissection doit être l'ultime solution car elle fait croître exponentiellement le besoin en ressource de calcul. Par contre, l'ajout de contraintes redondantes correspond parfaitement à la démarche d'un concepteur.
Comme exemple regardons comment spécifier un filtre analogique de type RLC. On spécifie la constante de temps τ du filtre liée à la fréquence centrale f_0 : $\tau = 1/(2\pi f_0)$. Cette spécification nous conduit à définir un intervalle de fréquence $[f]$ encadrant f_0. De $[f]$, on en déduit alors l'ensemble d'ajustements $\{R,L,C\}$ lié à la constante de temps τ.
La fréquence étant peu contrainte, il existe alors un grand nombre d'ajustements satisfaisant la spécification avec notamment R qui est un ajustement libre. Mais si on ajoute une spécification sur la bande passante BP avec un facteur d'amortissement : $m = BP/2f_0 > 5$, alors une contrainte supplémentaire est ajoutée sur l'ajustement R.
Augmenter les spécifications du filtre permet d'augmenter les contraintes sans enlever de point solution (sauf si les diverses spécifications sont incompatibles) mais sur contraindre le CSP réduit le risque de blocage des contracteurs. C'est donc naturel et utile.

5.6 Applications des opérateurs de contraction dans le cadre du dimensionnement de circuits

Le dimensionnement d'un circuit électrique consiste à fixer les valeurs de $\boldsymbol{n}$ variables d'ajustements $\boldsymbol{a} = (a_1, \ldots, a_n)^T$ de tel manière que les $\boldsymbol{m}$ performances $\boldsymbol{p} = (p_1, \ldots, p_m)^T$ du circuits satisfont TOUTES $\boldsymbol{m}$ spécifications $[S] = ([s_1], \ldots, [s_m])^T$. Ceci peut être formulé par un CSP de la forme :

$$\begin{aligned} p_1 &= f_1(a) \in [s_1] \\ p_2 &= f_2(a) \in [s_2] \\ &\ldots \\ p_m &= f_m(a) \in [s_m] \end{aligned}$$

avec

$$\begin{aligned} a_1 &\in [a_{c1}] \\ a_2 &\in [a_{c2}] \\ &\ldots \\ a_n &\in [a_{cn}] \end{aligned}$$

au final on peut donner un pavé de contraintes $[A_{Ctr}] = [a_{c1}] \times [a_{c2}] \times \ldots \times [a_{cn}]$

Ce pavé de contraintes représente les limites techniques ou technologiques. Il définit les hypothèses de travail qu'imposent par exemple des composants non-linéaires. A partir de ce pavé de contraintes, tous les points inconsistants seront rejetés hors de l'espace faisable. Ce rejet permet alors de limiter la durée de traitement.

En optant pour les écritures vectorielles, on ramène la formulation à :

$$\begin{aligned} \boldsymbol{p} &= \boldsymbol{f}(\boldsymbol{a}) \in [S] \\ \boldsymbol{a} &\in [A_{Ctr}] \end{aligned}$$

L'espace faisable est alors défini par

$$A_S = A_{Sf} \cap [A_{Ctr}] \text{ avec } A_{Sf} = \{ \boldsymbol{a} \in \mathbf{R}^n / \boldsymbol{f}(\boldsymbol{a}) \in [S] \} = \boldsymbol{f}^{-1}([S]).$$

La question que l'on doit poser au concepteur est alors :

veut-on trouver TOUT l'espace faisable, ou bien seulement une sous partie ?

Au final, le dimensionnement d'un circuit nécessite UN seul point, c'est à dire une valeur pour chaque variable d'ajustement. C'est pourquoi une approche ponctuelle est classiquement admise pour le dimensionnement. Pour extraire UN point de l'espace faisable, sachant que le concepteur aura souvent trop de degrés de liberté dans son système d'équations, on utilise un outil d'optimisation qui suivant sa fonction de coût trouvera UN point (local ou global) de fonctionnement satisfaisant le CSP.

Si l'objectif est de préparer le travail de l'outil d'optimisation, alors on peut encore restreindre l'espace de recherche à un sous pavé initial $[A]_{Init}$ (voir figure 5-8). Si celui-ci vérifie $[A]_{Init} \cap [A_{Ctr}] = \varnothing$, alors aucun point solution n'existe et l'optimiseur retournera une « erreur ».

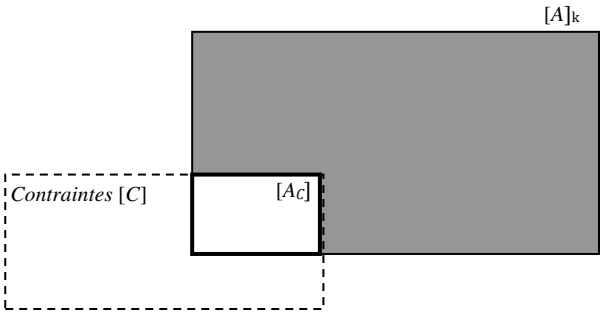


Figure 5-8 Réduction d'un pavé $[A]_k$ sous contraintes [C]

Si au contraire on souhaite explorer TOUT l'espace de conception, alors cela signifie que l'on veut trouver tout l'ensemble A_S. Cela impose donc que le pavé initial vérifie la condition suivante : $[A_{Ctr}] \subseteq [A]_{Init}$. Pour assurer cette condition, typiquement on fixe le pavé initial à $[A]_{Init} = \mathbf{R}^n$. Mais démarrer une analyse par intervalle pour approximer l'ensemble faisable A_S avec ce pavé initial conduit à un temps de calcul prohibitif. On préfère alors contracter ce pavé initial pour obtenir un premier pavé de travail $[A]_o$ à partir duquel un processus d'inversion ensembliste peut démarrer pour approximer par encadrement l'espace faisable A_S avec $[A]_o \equiv [A]_{Init} \cap [A_{Ctr}] \subseteq [A_{Ctr}]$. Utiliser pour cela un contracteur permet de garantir la relation suivante : $A_S = A_{Sf} \cap [A]_o \subseteq [A_{Ctr}]$.

L'utilisation des « *P-Opérateurs* » permet de construire un contracteur pour les fonctions performances. Dans chaque P-Opérateurs on retrouve le calcul direct $f([a])$ et le calcul inverse $f^{-1}([S])$ car pour les fonctions élémentaires l'expression de la fonction inverse existe.

Considérons un pavé $[A]_o$ qui correspond au sous-espace à réduire. L'image par f produit donc un pavé $[P]_o$ tel que : $[P]_o = [f]([A]_o)$. En procédant à une contraction locale (local au sens où l'on travaille sur une seule dimension) entre le pavé image $[P]_o$ et le pavé des spécifications $[S]$, on récupère un pavé image réduit :

$$[P_R]_o = [P]_o \cap [S]$$

Cette contraction locale ne perd aucun point dans l'espace des performances au regard du pavé initial. le pave réduit $[P_R]_0$ est ensuite rétro propagé dans l'espace des ajustements par f^{-1} tel que :

$$[A]_1 = [A]_0 \cap [f^{-1}]([P_R]_0)$$

Le cycle complet de contraction est illustré sur la figure 5-9.

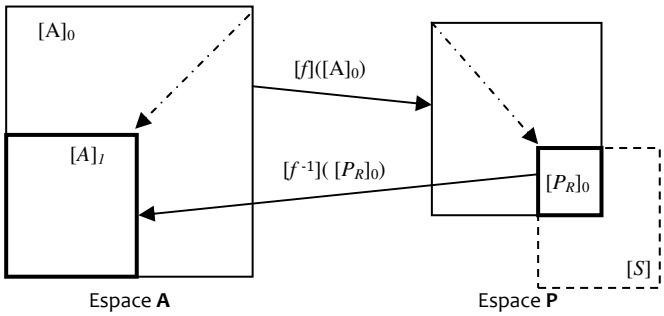


Figure 5-9 Contraction du pavé $[A]_0$ par la méthode de contraction par propagation entre l'espace des ajustements et l'espace des performances.

L'existence de f^{-1} est garantie ici par la décomposition en fonction élémentaire dont on connaît l'expression. Dans les faits, pour contracter une fonction f on utilise la décomposition en fonctions simples puis on introduit les P-Opérateurs.

Remarquons que la non existence d'une fonction élémentaire inverse n'interdit en rien l'utilisation de la méthode mais aucune contraction n'aura lieu sur cette composante. Cette direction devra donc subir une division, ceci pénalisera l'efficacité mais sans la bloquer.

5.7 Les stratégies d'exploration

Les paragraphes 5.3 et 5.4 ont présenté les approches par recouvrement et encadrement pour caractériser le bord d'un sous-espace de recherche A_S. Une approche différente a été présentée dans le paragraphe 5.5 et aborde l'exploration de l'espace par la résolution de CSP à l'aide des opérateurs de contraction.
Ces approches sont maintenant insérées dans un processus itératif faisant intervenir une opération de division, de classification et de contraction. L'opération de contraction est ponctuée d'une dernière opération de classification.
Cette stratégie d'exploration du sous-espace de recherche A_S est illustrée ci-dessous lorsqu'un pavé est en présence d'un point fixe.

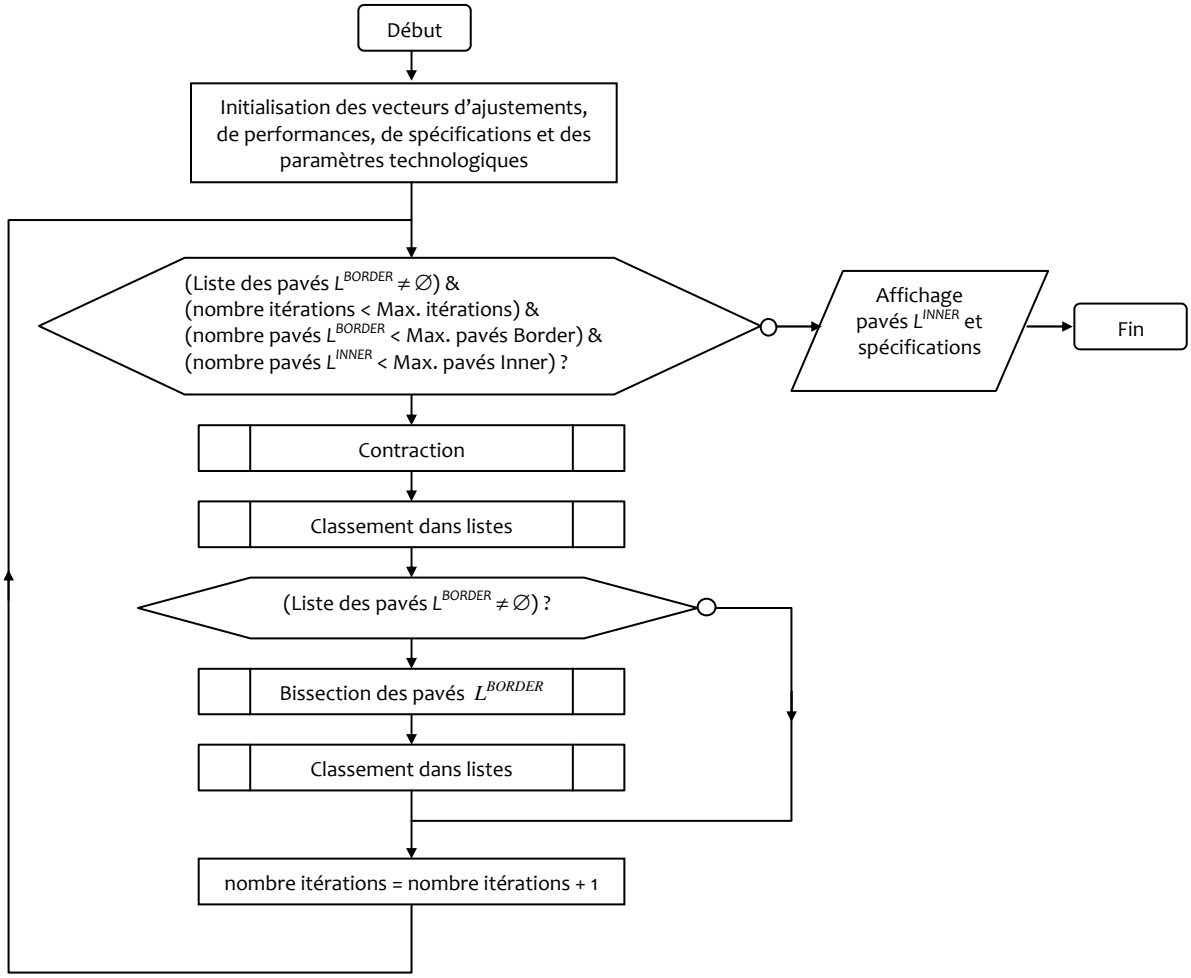
Début d'une itération de traitement d'un pavé (avec l'hypothèse de présence d'un point fixe)

① Division du pavé (bissection ou trisection)
② Classification des pavés obtenus après division.
On élimine les pavés dont on est « sûr » qu'ils appartiennent à la classe A^{INNER} ou $A^{OUTSIDE}$.
Les pavés indéterminés restants sont alors à retravailler : ils sont classés dans A^{BORDER}.
③ Contraction des pavés indéterminés (trois cas possibles sont à analyser)

Cas 1	Cas 2	Cas 3
Le pavé est proche de la frontière de A_S mais n'appartient pas à A_S.	Le pavé est proche de la frontière de A_S mais inclus dans A_S.	Le pavé contient la frontière de A_S
⇓	⇓	⇓
La contraction peut produire un pavé « vide » car inconsistant.	La contraction est sans effet sur le pavé.	Le pavé est contracté.
⇓	⇓	⇓
Ce pavé est classé dans $A^{OUTSIDE}$	Ce pavé est classé dans A^{INNER}	Ce pavé reste classé dans A^{BORDER}

Fin de l'itération

La stratégie que nous appliquerons est présentée sous la forme d'un organigramme. Les conditions d'arrêt du processus itératif d'exploration sont précisées en fonction du nombre d'itérations (limite de sécurité), du nombre de pavés admissibles classés dans la liste L^{BORDER} et dans la liste L^{INNER}.



Cette stratégie est maintenant appliquée en considérant un cas mathématique à vocation pédagogique. Dans cet exemple, les relations sont exprimées sans le moindre rapport formel avec des cas pratiques de l'électronique.

Nous définissons deux CSP H_1 et H_2 :

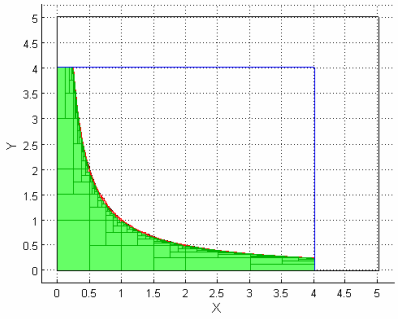
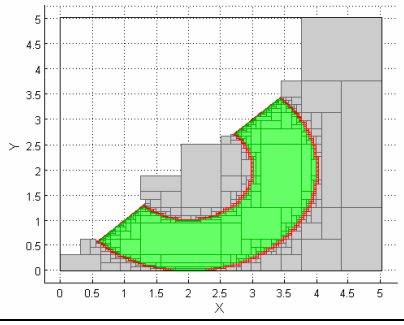
$$H_1 \begin{cases} g(x,y) = x.y \leq 1 \\ x \in [0,4] \\ y \in [0,4] \end{cases}$$

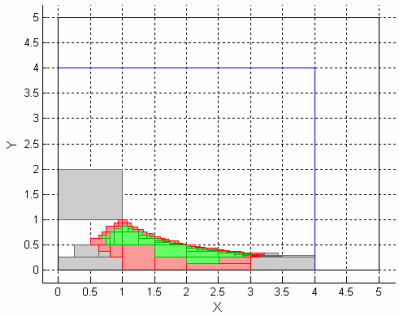
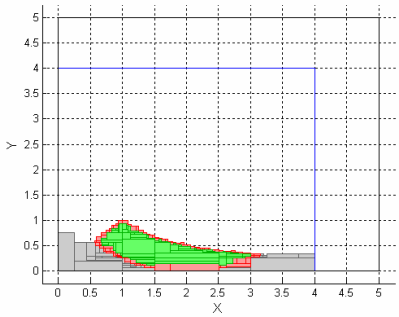
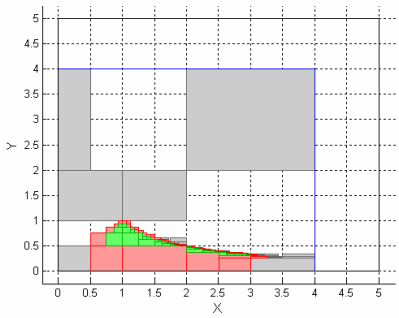
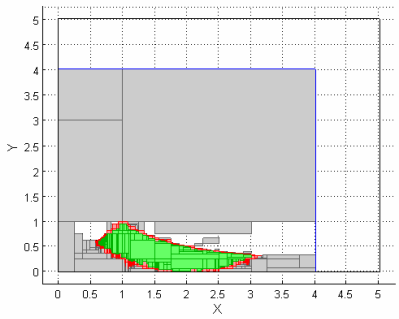
$$H_2 \begin{cases} f_1(x,y) = \sqrt{(x-2)^2 + (y-2)^2} \in [1,2] \\ f_2(x,y) = x - y \in [0,+\infty[\end{cases}$$

La recherche du sous-espace solution se limite au domaine de départ :

$$\begin{cases} x \in [0,5] \\ y \in [0,5] \end{cases}$$

Le tableau ci-dessous représente le pavage des classes A^{INNER}, A^{BORDER} et $A^{OUTSIDE}$ lorsque les CSP H_1 et H_2 sont traités séparément ou conjointement. Le processus de découpage associé est itératif ou systématique. Les pavés de la classe A^{INNER} sont représentés en vert, les pavés de la classe A^{BORDER} en rouges. Les pavés de la classe $A^{OUTSIDE}$ sont grisés en précisant qu'à l'issue d'une opération de contraction, un pavé indéterminé et inconsistant produit un pavé « vide ». Ce pavé vide échappe au processus de classification et aucun marquage par la couleur n'est possible. Ce pavé correspond sur l'illustration à un sous-espace vide de couleur.

Stratégie de recherche	**Pavage des listes A^{INNER}, A^{BORDER} et $A^{OUTSIDE}$**	**Résultats**
① Résolution de H_1. Découpage itératif par bissection.		Temps d'exécution : 4.94s Nombre d'itérations : 13 Nombre de pavés inner : 540 Nombre de pavés border : 916 Nombre de pavés outside : 0
② Résolution de H_2. Découpage itératif par bissection.		Temps d'exécution : 6.13s Nombre d'itérations : 13 Nombre de pavés inner : 423 Nombre de pavés border : 787 Nombre de pavés outside : 223

③ Résolution de $H = H_1 \cap H_2$. Découpage itératif par bissection.		Temps d'exécution : 4.76s Nombre d'itérations : 19 Nombre de pavés inner : 37 Nombre de pavés border : 67 Nombre de pavés outside : 13
④ Résolution de $H = H_1 \cap H_2$. Découpage itératif par trisection.		Temps d'exécution : 6.26s Nombre d'itérations : 19 Nombre de pavés inner : 71 Nombre de pavés border : 108 Nombre de pavés outside : 44
⑤ Résolution de $H = H_1 \cap H_2$. Découpage systématique par bissection.		Temps d'exécution : 5.26s Nombre d'itérations : 19 Nombre de pavés inner : 34 Nombre de pavés border : 61 Nombre de pavés outside : 23
⑥ Résolution de $H = H_1 \cap H_2$. Découpage systématique par trisection.		Temps d'exécution : 17.20s Nombre d'itérations : 12 Nombre de pavés inner : 432 Nombre de pavés border : 795 Nombre de pavés outside : 331

La résolution des CSP H_1 et H_2 a été traitée selon six stratégies différentes. La première stratégie propose la résolution de H_1 uniquement. L'espace solution A_S est alors constitué des pavés placés sous la courbe $g(x,y)$. Ces pavés forment la classe A^{INNER}. Les pavés qui ne sont pas situés dans A_S et sur sa frontière sont rejetés par contraction et échappent à la classification. La classe $A^{OUTSIDE}$ reste donc vide.

La seconde stratégie applique le même procédé de résolution du CSP H_2. La représentation de l'équation du cercle $f_1(x,y)$ ressort nettement avec un plan de coupe imposé par la seconde équation $f_2(x,y)$. Cette configuration fait apparaître un ensemble de pavés de la classe $A^{OUTSIDE}$ qui résultent d'un classement après division.

La troisième stratégie fournit un pavage de l'espace solution commun (l'intersection) généré par la résolution de H_1 et H_2. Des pavés indéterminés de taille importante subsistent et induisent une

surestimation du profil inférieur de la région solution. Cette approximation est ensuite réduite en préférant le découpage par trisection au découpage par bissection (stratégie N°4). Par ce découpage plus soutenu (trois sous-pavés sont produits après trisection), davantage de sous pavés sont soumis au test de classification et classés dans les listes.
Enfin, les deux dernières stratégies (découpage systématique) n'apportent que très peu d'amélioration en terme de précision. Un nombre important de pavés est cependant généré très tôt dans le processus itératif tenant à la nature du découpage. Le partage de l'espace est alors traité de manière systématique dans toutes les directions et avec un temps d'exploration devenant prépondérant.

5.8 Conclusion

Nous avons introduit une première approche de conception appliquée au pré-dimensionnement d'un circuit électronique. Cette approche s'appuie sur les techniques de l'analyse par intervalles. Elle permet de résoudre un CSP afin de préparer au mieux la réussite de l'étape de l'optimisation en proposant un sous-espace d'ajustements compatible avec l'ensemble de toutes les spécifications.
Dans ce cadre, le dimensionnement d'un circuit consiste à explorer l'espace des ajustements par le concept de recouvrement par pavage. Ce concept peut être appliqué à la caractérisation d'un sous-espace des ajustements en abordant l'approche par inversion ensembliste. L'approche de l'inversion ensembliste repose sur la notion de fonctions d'inclusions naturelles qui permet de construire un pavé image dans l'espace des performances. L'objectif étant de transposer chaque pavé du sous-espace des ajustements dans l'espace des performances et de vérifier son appartenance au pavé des spécifications. Cette approche permet donc bien d'associer un sous-espace de l'espace des performances avec son sous-espace antécédent dans l'espace des ajustements. De cette association résulte le classement des pavés d'ajustement en trois sous-ensembles. Un pavé des ajustements appartient au sous-ensemble $A^{OUTSIDE}$ si son image par la fonction d'inclusion est à l'extérieur du pavé des spécifications. Si en revanche le pavé image est totalement inclus dans le pavé des spécifications, on a alors la garantie que tous les points du pavé des ajustements ont les images dans le pavé des spécifications. Ce pavé des ajustements appartient alors au sous-ensemble des pavés intérieurs A^{INNER}. Si la comparaison de l'image d'un pavé des ajustements au pavé des spécifications ne permet pas de statuer clairement sur son appartenance, le pavé des ajustements appartient alors au sous-ensemble A^{BORDER}. La réunion des pavés du sous-ensemble A^{INNER} permet alors de construire une approximation du sous-espace des ajustements. Le niveau d'approximation est déterminé par un processus itératif de division des pavés indéterminés appartenant au sous-ensemble A^{BORDER} en sous-pavés. Le traitement de ces sous-pavés permet alors d'approcher la frontière du sous-espace des ajustements. Un tel processus nécessite en contrepartie d'importantes ressources en calculs qui augmentent avec la dimension du problème à traiter. Il importe donc de limiter la taille des pavés indéterminés avant leur division. Or, l'utilisation de fonctions d'inclusions naturelles introduit une surestimation du pavé image par rapport à son image exacte. Cette surestimation conduit à un classement plus systématique des pavés d'ajustement dans l'ensemble A^{BORDER}.
Le recourt aux opérateurs de « réduction » (narrowing operators) encore appelés les opérateurs de « contraction » (contractors) permettent de rejeter immédiatement un sous-espace de points des ajustements pour inconsistance avec les spécifications. Les opérateurs de contraction sont remarquables pour leur efficacité à traiter des pavés de grande taille notamment dans des espaces à grandes dimensions (supérieures à 10). Ils permettent de décomposer une expression analytique en une série d'opérations élémentaires qu'un algorithme pourra traiter. Le traitement séquentiel à travers ces opérations élémentaires introduit une propagation de l'information qui aboutit à l'expression d'un pavé minimal sans qu'aucune division ne soit nécessaire. L'utilisation des opérateurs de contraction permet d'améliorer l'exploration de l'espace de conception en

réduisant le temps de calcul sur les pavés indéterminés qui définissent la frontière de l'espace des ajustements.

Dans le prochain chapitre, nous appliquerons ces méthodes sur des cas pratiques et en relation avec le support de notre étude (réseau de neurones). Nous verrons notamment que le caractère majorant des méthodes présentées n'implique pas forcément l'appel d'un processus long de pavage pour réduire l'effet de surapproximation. Une stratégie de recherche basée sur les méthodes par contraction et d'inversion ensembliste sera proposée. Nous pourrons alors mettre en oeuvre une méthode complémentaire qui permettra d'estimer le volume de l'espace de faisabilité qui résulte des méthodes d'exploration d'espace que nous venons de présenter. Ce volume nous permettra de mesurer la robustesse d'une architecture particulière aux risques de dérives liées à sa fabrication.

Chapitre 6 - application de l'analyse par intervalles au dimensionnement de circuits analogiques

6.1 Introduction

Le dimensionnement d'un circuit analogique nécessite de déterminer avant tout un point de fonctionnement autours duquel les signaux électriques pourront varier au cours du temps. Dans une hypothèse de petites variations (approche « petit signal »), le circuit peut être linéarisé et une fonction de transfert décrivant la fonctionnalité du circuit peut être établie. Or les caractéristiques de cette fonctionnalité (que nous appellerons « performances ») sont essentiellement conditionnées par le dimensionnement du circuit. Celui-ci détermine la valeur nominale des composants du circuit, valeurs qui déterminent à leur tour le point de fonctionnement nominal.
Lors de la fabrication du circuit dimensionné, les valeurs réalisées s'écarteront des valeurs nominales. On définira alors la robustesse d'un circuit par rapport à la production du circuit comme la plage de valeurs que peut prendre chaque variable indépendamment l'une de l'autre tout en maintenant les caractéristiques du circuit dans une plage acceptable. Cette plage acceptable correspond aux spécifications des performances du circuit. Plus les plages des variables à dimensionner (les variables « d'ajustements » pour nous) sera grande, plus le circuit sera robuste face à la fabrication.
Parler de plage de valeurs revient naturellement à parler d'intervalles. On comprendra immédiatement que l'ensemble des plages des variables d'ajustement formera un pavé [A] dans l'espace des ajustements, que l'ensemble des plages acceptables des performances formera un pavé [S] dans l'espace des performances. Le dimensionnement consiste donc partant d'un pavé de spécifications [S] à trouver UN pavé des ajustements [A_{rob}] le plus large possible tel que tous les points de ce pavé auront une image dans [S]. Selon cet énoncé, le dimensionnement est un problème inverse dans un contexte d'espaces cartésiens. On parle alors d'exploration de l'espace de conception.
Une autre évidence découle de cet énoncé : réduire le pavé [S] induira une réduction du pavé [A_R]. Si la procédure d'extraction d'un pavé [A_{rob}] est relativement rapide, alors on pourra envisager de « tester » le pavé de spécifications [S]. On peut envisager deux cas de figure qui nous semblent intéressants. Par une première évaluation, on montre qu'un pavé [A_{rob}] compatible avec une technologie existe. On peut alors décider de réduire les spécifications de ce circuit pour relâcher d'autres spécifications dans le système global de l'application. Dans le second cas, on peut soumettre le pavé [S] à différents circuits (différentes topologies) de même fonctionnalité et comparer les pavés résultants [A_{rob}]. De cette comparaison on peut extraire le circuit le plus robuste et opérer ainsi une sélection topologique. Ce dernier point n'est pas abordé dans cette thèse mais fut un objectif latent dans notre démarche.
Les divers objectifs que nous avons pointés ci-dessus doivent être atteints dans un temps et avec une précision raisonnable. C'est pourquoi nous pensons que les méthodes classiques d'analyse numériques (approche ponctuelle) ne peuvent couvrir l'exploration de l'espace de conception de manière efficace en donnant avec garanti TOUS les points faisables. L'introduction des calculs par intervalles veut offrir une réponse acceptable à ces objectifs. Ce travail de thèse n'est qu'une première tentative (du moins à notre connaissance et à ce jour) et pourra encore être largement amélioré compte tenu du fait que nous avons utilisé les solutions les plus simples. Or de nouvelles techniques d'analyses par intervalles sont publiés régulièrement et offre des améliorations parfois substantielles par rapport aux techniques de base.
Nous commencerons la présentation de l'application de ces techniques par un exemple académique dont on trouve les résultats d'une analyse classique dans [Joh-97] afin de pouvoir les comparer à nos résultats. Cet exemple restera extrêmement simplifié afin de poser clairement les étapes de notre démarche de dimensionnement. Le second exemple utilise un circuit extrêmement classique en électronique analogique. Il est un élément typique dans nos circuits

neuronaux mais aussi une référence dans les publications portant sur le dimensionnement. Ce dernier point est important afin de pouvoir établir une comparaison plus large avec d'autres approches, mais regroupe aussi divers problèmes typiques comme par exemple l'impact de la robustesse de production sur le dimensionnement.

Nous donnerons les temps d'exécution des algorithmes (des processus de calculs) écrit sous Matlab et fonctionnant sur un ordinateur de type PC avec un processeur Intel Pentium Centrino cadencé à 1.6 GHz et 1 Giga Octet de mémoire vive.

6.2 Première application à un circuit académique

6.2.1 Présentation du circuit et des objectifs de dimensionnement

Nous avons appliqué nos méthodes de dimensionnement à un récepteur de fibre optique. Le récepteur est formé d'un amplificateur de transimpédance dont le premier étage est constitué un amplificateur CMOS en Source Commune [Joh-97]. Le schéma est rappelé sur la figure 6-1.

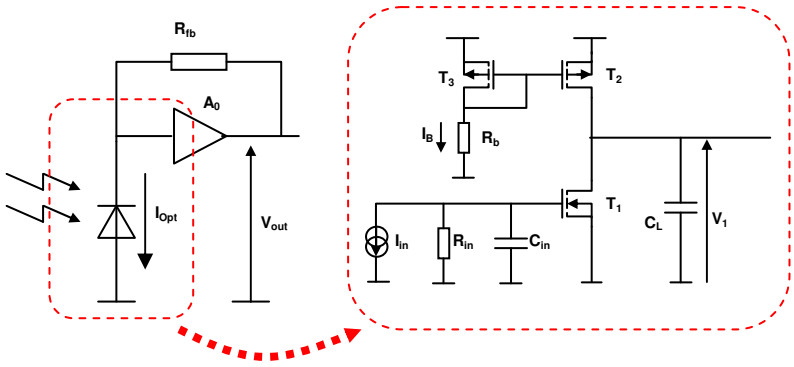


Figure 6-10 Schéma de l'amplificateur de transimpédance

Nous nous focalisons sur le dimensionnement des transistors T_1 et T_2. Le transistor T_3 pouvant se déduire du transistor T_2 par le rapport de transformation du miroir de courant. Les objectifs de conceptions sont :
la tension de sortie doit être médiane,
le gain en tension doit être maîtrisé (assez élevé pour sa contribution au gain total mais sans excès pour la stabilité)
l'impédance de sortie de l'amplificateur doit permettre de réduire les temps de réponses du récepteur (débit de transmission).

Lors du dimensionnement deux aspects complémentaires doivent être gérés :
les régimes de fonctionnements des composants,
les caractéristiques liées à la fonctionnalité.

La fonctionnalité du circuit repose sur une hypothèse souvent non exprimé car évidente, c'est que les transistors MOS T_1 et T_2 doivent impérativement travailler en régime de forte inversion. Pour cela on doit contrôler la tension V_{DSsat}. Les équations suivantes doivent donc être vérifiées :

$$V_{DSsat} \geq V_{DSsat\,min} = V_{GS} - V_{TH} = 0.1V \qquad (1),$$

$$I_B = \frac{K_n}{2}\frac{W_1}{L_1}V_{DSsat}^{\ 2} \qquad (2),$$

$$I_B = \frac{K_p}{2}\frac{W_2}{L_2}V_{DSsat}{}^2 \qquad (3).$$

L'objectif de positionner la tension de sortie au milieu de la dynamique impose les conditions suivantes : $V_{DSsat1} = V_{DSsat2} = V_{DSsat}$. Ce qui conduit à $V_{DD} = V_{DSsat1} + V_{DSsat2} = 2.V_{DSsat}$, soit enfin :
$V_{DD} = 2.V_{DSsat}$ (4),

De fait la variable $(W/L)_2$ est liée à la variable $(W/L)_1$ par : $(W/L)_2 = K_n/K_p * (W/L)_1$. Ce qui transforme les équations 5, 6 et 7 en :

$V_{DD} = 2.V_{DSsat}$ (5),
$$I_B = \frac{K_n}{2}\frac{W_1}{L_1}V_{DSsat}{}^2 \qquad (6),$$
$(W/L)_2 = (K_n/K_p).(W/L)_1$ (7),

sous les conditions :

$V_{DSsat} \geq V_{DSsat\,min} = V_{GS} - V_{TH} = 0.1V$ (8),
$1V \leq V_{DD} \leq 5V$ (9),

Les caractéristiques fonctionnelles sont l'impédance de sortie R_{OUT} et le gain en tension A_{V1} de l'étage. Ces caractéristiques sont pour nous des performances et sont exprimées par :

$$\left|A_{V1}\right| = g_{m1}\cdot R_{Out} = f_1\left(I_B, \left(\frac{W}{L}\right)_1, L_1, L_2\right), \qquad (10)$$

$$R_{Out} = \frac{R_{ds1}\cdot R_{ds2}}{R_{ds1}+R_{ds2}} = f_2\left(I_B, L_1, L_2\right) \qquad (11)$$

Les paramètres internes à ces deux expressions sont explicités ci-dessous :

$$g_{m1} = \sqrt{\mu_N C_{OX}\left(\frac{W}{L}\right)_1 I_B}\,, \qquad (12)$$

$$R_{ds1} = \frac{\rho_n.L_1}{I_B}, \quad R_{ds2} = \frac{\rho_p.L_2}{I_B}, \qquad (13).$$

Les spécifications qui fixent les plages acceptables de ces caractéristiques sont:

$[Sp] = [S\_A_{V1}] \times [S\_R_{Out}] =$ [67.5, 162.5].$10^3 \times$ [450, 650]kΩ.

En examinant la relation (10), il vient que la spécification $[S\_A_{V1}]$ peut être transposée en une spécification $[S\_g_{m1}]$ caractérisant la transconductance g_{m1} du transistor T_1. Son intervalle étant strictement positif, sa transposition est donc triviale et revient à reconsidérer la nouvelle spécification par la simple expression arithmétique : $S(g_{m1}) = S(A_{V1})/S(R_{Out})$. Par conséquent, en rendant la performance g_{m1} compatible avec sa spécification $[S\_A_{V1}]$, on fait correspondre mécaniquement le gain en tension A_{V1} avec sa propre spécification $[S\_A_{V1}]$. Les nouvelles spécifications s'exprime en définitive par :

$[Sp^*] = [S\_g_{m1}] \times [S\_R_{Out}] =$ [150, 250]µA/V × [450, 650]kΩ.

Enfin, les paramètres technologiques du circuit sont considéré ici comme fixe et égales à :

ρ_n = 8e6 ΩA/m, ρ_p = 12e6 ΩA/m, K_n = 90 µA/V², K_p = 22 µA/V²,
R_{fb}=180kΩ, C_{in} = 0,2pF, C_{gd1} = 0,015pF et C_L = 0,4pF.

Dans une analyse plus complète (et aussi plus lourde en besoin de calcul) ces paramètres peuvent être transformés en intervalles fixes. C'est à dire qu'ils interviennent dans les équations comme intervalles mais qu'ils ne peuvent être modifiés par les calculs.

Toujours dans un souci de simplification de l'exposé, nous allons réduire le nombre de variables indépendantes ce qui facilitera la construction et l'illustration des images du sous espace faisable. Pour cela nous fixons la longueur L_1 du canal du transistor T_1 à : L_1 = 1.6µm. Ce choix permet d'assurer une « tension Early » suffisamment importante afin que T_1 travaille effectivement comme une source de courant.

L'exploration de l'espace de conception sera également réduite arbitrairement. Les valeurs qui suivent s'accordent avec une technologie de 0.18µm. On se donne comme domaine d'analyse pour les variables d'ajustement les plages suivantes :

- le courant de polarisation sera borné par : $0.01 \leq I_B \leq 50\mu A$;
- le rapport $(W/L)_1$ sera borné par : $0.1 \leq (W/L)_1 \leq 100$;
- la longueur L_2 du canal du transistor T_2 sera bornée par : $0.1 \leq L_2 \leq 3\mu m$;

Le problème du dimensionnement peut alors se présenter sous la forme d'un CSP *H*. Mais pour en faciliter la compréhension et mettre en avant les transformations nécessaires, ce problème sera décomposé en deux CSP H_1 et H_2.

Le CSP H_1 porte sur les conditions à satisfaire par les composants du circuit (nous parlons de « *Device Behavior Performances* »). Les fonctions du CSP H_1 sont :

$$f_{DB1}(a_{DB}) : I_B = \frac{1}{2} Kn \frac{W_1}{L_1} V_{DSsat}{}^2, \qquad (14)$$

$$f_{DB2}(a_{DB}) : I_B = \frac{1}{2} Kp \frac{W_2}{L_2} V_{DSsat}{}^2, \qquad (15)$$

$$f_{DB3}(a_{DB}) : V_{DD} = 2.V_{DSsat}, \qquad (16)$$

avec les contraintes limitant la plage d'analyse :

$$a_{DB} \in [A_{DB}] = [I_B] \times [(W/L)_1] \times [V_{DSsat}] \times [V_{DD}] = [0.01, 50] \times [0.1, 100] \times [0.1, 10^9] \times [1, 5].$$

Ce CSP H_1 est alors de la forme :

$f_{DB}(a_{DB}) = 0$,
$a_{DB} \in [A_{DB}]$.

Le CSP H_2 porte sur les conditions à satisfaire par les caractéristiques fonctionnelles. Nous parlerons ici simplement de performances sans préciser qu'elles sont liées au fonctionnement du circuit. Les fonctions du CSP H_2 sont :

$$f_{P1}(a_P): R_{Out} = \frac{R_{ds1} \cdot R_{ds2}}{R_{ds1} + R_{ds2}}, \quad (17)$$

$$f_{P2}(a_P): g_{m1}{}^2 = \mu_m C_{Ox}.I_B.(W/L)_1, \quad (18)$$

avec les contraintes limitant la plage d'analyse :

$a_P \in [A_P] = [I_B]\times[(W/L)_1]\times[L_2] = [0.01, 50]\times[0.1, 100]\times[0.1, 3].$

Ce CSP H_2 est alors de la forme :

$f_P(a_P) \in [S_P{}^*]$,
$a_P \in [A_P]$.

Par les transformations que nous avons présentées chapitre 4, le CSP H_1 peut alors être changé en CSP H_1***** s'écrivant :

$$f_{DB1}(a_{DB}): I_B = \frac{1}{2} Kn \frac{W_1}{L_1} V_{DSsat}{}^2 = 45 \frac{W_1}{L_1} V_{DSsat}{}^2, \quad (19)$$

$$f_{DB2}(a_{DB}): I_B = \frac{1}{2} Kn \frac{W_2}{L_2} V_{DSsat}{}^2 = 11 \frac{W_2}{L_2} V_{DSsat}{}^2, \quad (20)$$

$$f_{DB3}(a_{DB}): V_{DSsat} = u, \quad (21)$$

$$f_{DB4}(a_{DB}): V_{DD} = 2.u, \quad (22)$$

avec les contraintes limitant la plage d'analyse :

$a_{DB} \in [A_P]$ et
$u \in [0, +\infty]$ une variable auxiliaire.

En définissant le pavé des spécifications des composants :

$[S_{DB}] = [V_{DSsat}]\times[V_{DD}] = [0.1, 10^9]\times[1, 5]$

Ce CSP H_1***** est alors de la forme :

$f_{DB}(a_{DB}) \in [S_{DB}]$,
$a_{DB} \in [A_P]$.

Ainsi le CSP H global devra vérifier $H = H_1^* \cap H_2$ et s'exprimera par :

$f_{DB}(a) \in [S_{DB}]$,
$f_P(a) \in [S_P{}^*]$,
$a \in [A] = [I_B]\times[(W/L)_1]\times[L_2] = [0.01, 50]\times[0.1, 100]\times[0.1, 3].$

*6.2.2 Résolution partielle du CSP des composants : H_1**

Résoudre un CSP c'est trouver l'ensemble de points appartenant à l'ensemble faisable A_S (*cf.* §4.5.1). Mais comme préalable à la description fine et détaillé de l'ensemble faisable, on peut rechercher le plus petit pavé contenant l'ensemble faisable A_S. Dans cette optique, les contracteurs de type *pOperators* offrent une méthode efficace pour s'approcher de ce plus petit

pavé (*cf.* chapitre 5, figure 5-7). Trouver un pavé contracté correspond pour nous à une résolution partielle du CSP !

Comme résoudre le CSP global passe par la résolution des Sous-CSP, nous utiliserons cette approche. En fait, résoudre en premier le CSP H_1* permet de vérifier si le pavé de recherche [*A*] est compatible avec les hypothèses de fonctionnement des composants. Si le sous-espace faisable de ce CSP est vide on sait immédiatement que l'espace faisable du CSP *H* est vide.

Pour opérer la contraction du domaine de recherche [*A*], nous transposons les équations $f_{DB}(a)$ avec une suite d'opérations élémentaires où l'on introduit les *pOpérateurs*. Après contraction du domaine de recherche [*A*] on récupère le pavé [A_{C1}].

Chacune des expressions de (19) à (22) sont décomposées en primitives et exécutées récursivement. Le mécanisme de rétropropagation est détaillé dans le tableau 6-1.

Etape	Primitives (pOperators)	Fonction associée	Expression contractée
1	$[u] := [u] \cap ([V_{DSsat}]*[V_{DSsat}])$	$u = V_{DSsat}^2$	$f_{DB1}(a_{DB})$
	$[V_{DSsat}] := [V_{DSsat}] \cap sqrt[u]$		
2	$[I_B] := [I_B] \cap (45*[W_1/L_1]*[u])$	$I_B = 45*(W_1/L_1)* u$	
	$[W_1/L_1] := [W_1/L_1] \cap ((1/45)*[I_B]/[u])$		
	$[u] := [u] \cap ((1/45)*[I_B]/[W_1/L_1])$		
3	$[u] := [u] \cap ([I_B]/[45*W_1/L_1])$	$u = I_B/(45*W_1/L_1)$	
	$[I_B] := [I_B] \cap ([45*W_1/L_1]*[u])$		
	$[W_1/L_1] := [W_1/L_1] \cap ([I_B]/[45*u])$		
4	$[V_{DSsat}] := [V_{DSsat}] \cap sqrt[u]$	$V_{DSsat} = sqrt(u)$	
	$[u] := [u] \cap ([V_{DSsat}]*[V_{DSsat}])$		
5	$[u] := [u] \cap ([V_{DSsat}]*[V_{DSsat}])$	$u = V_{DSsat}^2$	
	$[V_{DSsat}] := [V_{DSsat}] \cap sqrt[u]$		
6	$[W_1/L_1] := [W_1/L_1] \cap ([I_B]/[45*u])$	$W_1/L_1 = I_B/(45*u)$	
	$[I_B] := [I_B] \cap ([45*W_1/L_1]*[u])$		
	$[u] := [u] \cap ([I_B]/[45*W_1/L_1])$		
7	$[u] := [u] \cap ([V_{DSsat}]*[V_{DSsat}])$	$u = V_{DSsat}^2$	$f_{DB2}(a_{DB})$
	$[V_{DSsat}] := [V_{DSsat}] \cap sqrt[u]$		
8	$[I_B] := [I_B] \cap (11*[W_2/L_2]*[u])$	$I_B = 11*(W_2/L_2)* u$	
	$[W_2/L_2] := [W_2/L_2] \cap ((1/11)*[I_B]/[u])$		
	$[u] := [u] \cap ((1/11)*[I_B]/[W_2/L_2])$		
9	$[u] := [u] \cap ([I_B]/[11*W_2/L_2])$	$u = I_B/(11*W_2/L_2)$	
	$[I_B] := [I_B] \cap ([11*W_2/L_2]*[u])$		
	$[W_2/L_2] := [W_2/L_2] \cap ([I_B]/[11*u])$		
10	$[V_{DSsat}] := [V_{DSsat}] \cap sqrt[u]$	$V_{DSsat} = sqrt(u)$	
	$[u] := [u] \cap ([V_{DSsat}]*[V_{DSsat}])$		
11	$[u] := [u] \cap ([V_{DSsat}]*[V_{DSsat}])$	$u = V_{DSsat}^2$	
	$[V_{DSsat}] := [V_{DSsat}] \cap sqrt[u]$		
12	$[W_2/L_2] := [W_2/L_2] \cap ([I_B]/[11*u])$	$W_2/L_2 = I_B/(11*u)$	
	$[I_B] := [I_B] \cap ([11*W_2/L_2]*[u])$		
	$[u] := [u] \cap ([I_B]/[11*W_2/L_2])$		
13	$[V_{DD}] := [V_{DD}] \cap ([u]+[u])$	$V_{DD} = u+u$	$f_{DB4}(a_{DB})$
	$[u] := [u] \cap ([V_{DD}]-[u])$		
	$[u] := [u] \cap ([V_{DD}] - [u])$		

Tableau 6-6 Décomposition des expressions (19) à (22) en primitives (pOperators)

Le processus de contraction converge en 340ms après trois itérations et le sous espace $[A_{C1}]$ est donné par:

$[A_{C1}] = C_{H1*}([A],[S_{DB}])$
$[A_{C1}] = [I_B] \times [(W/L)_1] \times [L_2]$
$[A_{C1}] = [0.4, 49.9] \times [0.1, 27.7] \times [0.1, 3].$

On constatera que la contraction a laissé l'intervalle $[L_2]$ intact puisqu'il n'y avait aucune condition sur cette variable. Par contre le processus a contracté les deux autres variables d'ajustement : I_B et $(W/L)_1$.

6.2.3 Résolution partielle du CSP des fonctionnalités : H_2

Après avoir résolu partiellement le CSP H_1*, nous allons faire de même avec le CSP H_2 portant sur les caractéristiques fonctionnelles. Mais pour gagner du temps au lieu de démarrer la contraction du pavé de recherche $[A]$, nous partons directement du pavé $[A_{C1}]$ car les solutions du CSP global H devrons aussi satisfaire H_1* et donc les points faisable au global devrons appartenir à ce pavé.

Les équations de H_2 sont transformées en une suite d'opérateurs de contractions du type P-Opérateur et le pavé résultant sera $[A_{C2}]$.

Etape	Primitives (pOperators)	Fonction associée	Expression contractée
1	$[R_{ds1}] := [R_{ds1}] \cap (8e3*[L_1]/[I_B])$	$R_{ds1} = 8e3*(L_1/I_B)$	$f_{P1}(a_P)$
	$[L_1] := [L_1] \cap ([1/8e3*R_{ds1}]*[I_B])$		
	$[I_B] := [I_B] \cap (8e3*[L_1]/[R_{ds1}])$		
2	$[R_{ds2}] := [R_{ds2}] \cap (12e3*[L_2]/[I_B])$	$R_{ds2} = 12e3*(L_2/I_B)$	
	$[L_2] := [L_2] \cap ([1/12e3*R_{ds2}]*[I_B])$		
	$[I_B] := [I_B] \cap (12e3*[L_2]/[R_{ds2}])$		
3	$[u_1] := [u_1] \cap ([R_{ds1}]*[R_{ds2}])$	$u_1 = R_{ds1}*R_{ds2}$	
	$[R_{ds1}] := [R_{ds1}] \cap ([u_1]/[R_{ds2}])$		
	$[R_{ds2}] := [R_{ds2}] \cap ([u_1]/[R_{ds1}])$		
4	$[u_2] := [u_2] \cap ([R_{ds1}]+[R_{ds2}])$	$u_2 = R_{ds1} + R_{ds2}$	
	$[R_{ds1}] := [R_{ds1}] \cap ([u_2]-[R_{ds2}])$		
	$[R_{ds2}] := [R_{ds2}] \cap ([u_2]-[R_{ds1}])$		
5	$[R_{out}] := [R_{out}] \cap ([u_1]/[u_2])$	$R_{out} = (u_1/u_2)$	
	$[u_1] := [u_1] \cap ([R_{out}]*[u_2])$		
	$[u_2] := [u_2] \cap ([R_{out}]*[u_1])$		
6	$[u_3] := [u_3] \cap (90*[W_1/L_1]*[I_B])$	$u_3 = 90*(W_1/L_1)*I_B$	$f_{P2}(a_P)$
	$[W_1/L_1] := [W_1/L_1] \cap (1/90*[u_3]/[I_B])$		
	$[I_B] := [I_B] \cap (1/90*[u_3]/[W_1/L_1])$		
7	$[g_{m1}] := [g_{m1}] \cap sqrt[u_3]$	$g_{m1} = sqrt(u_3)$	
	$[u_3] := [u_3] \cap ([g_{m1}]*[g_{m1}])$		

Tableau 6-7 Décomposition des expressions (17) à (18) en primitives (pOperators)

Après 10 itérations, soit une durée de calcul de 480ms, le processus retourne le sous espace contracté $[A_{C2}]$:

$[A_{C2}] = C_{H2}([A_{C1}],[S_P])$
$[A_{C2}] = [I_B] \times [(W/L)_1] \times [L_2]$
$[A_{C2}] = [4.5, 48.9] \times [2.5, 27.7] \times [0.1, 3].$

Du fait de démarrer la résolution partielle du CSP H_2 fait que le pavé contracté $[A_{C2}]$ sera immédiatement le pavé contracté pour le CSP H.

Les opérations de contractions ont permis de réduire l'espace de recherche en éliminant les zones inconsistantes avec les contraintes du problème. On observe cependant que les contracteurs n'ont pas réussi à réduire le pavé recherche selon toutes les dimensions. Ceci peut avoir pour origine soit un problème de point fixe pour les contracteurs, soit provenir de la surestimation introduite par les fonctions d'inclusions. Ce dernier point peut être soit réglé si l'on ré-écrit les équations en limitant les multi occurrences des variables ou bien en améliorant les fonctions d'inclusions. Sur ce dernier point, les contracteurs peuvent aussi être utilisés comme le montre [Jau-01b]. Il existe d'autres pistes pour améliorer cette situation que nous n'avons pu explorer pour l'instant.

6.2.4 Exploration de l'espace de « conception » et description de l'espace faisable

Les opérations de contraction nous ont réduit l'espace dans lequel se trouve l'ensemble solution A_S. Même si le pavé contracté est le plus petit pavé contenant l'ensemble A_S, il contient encore un grand nombre de points n'appartenant pas à A_S. Pour affiner notre approximation de l'espace faisable A_S, nous allons utiliser la méthode par encadrement qui met en œuvre l'inversion ensembliste. Nous démarrerons cet encadrement à partir du pavé $[A_{C2}]$.

Cette méthode peut bien entendu être appliquée à n'importe quel stade de la phase d'exploration et notamment sans recourir aux contractions initiales. Cela pourrait par ailleurs être envisagé si aucune hypothèse sur les composants n'est disponible. Mais il est nécessaire de rappeler et les résultats présentés plus loin dans cet exemple le démontreront encore, que la méthode par encadrement nécessite des ressources mémoire importantes et le fait de procéder à un encadrement d'un espace large et non contracté entraîne un temps d'exécution incompatible avec les attentes actuelles.

La méthode par encadrement, nous l'avons vu, exploite le découpage de l'espace des ajustements afin de pouvoir opérer un test de classification des pavés de performances relativement au pavé de spécifications. Par soucis de simplicité, dans cet exemple, le découpage est ici systématique. Chaque dimension a été bissecté six fois. L'espace d'ajustement comporte alors $(2^3)^6 = 8^6 = 262144$ pavés au total après division. Chaque pavé élémentaire sera donc projeté dans l'espace des performances et subira le test de classification. Ce processus a classé 5240 pavés comme pavés d'intérieurs dans l'ensemble $\boldsymbol{A}^{INNER}$, 18690 pavés comme pavés de bord dans l'ensemble $\boldsymbol{A}^{BORDER}$ et 238214 pavés comme pavé extérieur dans l'ensemble $\boldsymbol{A}^{OUTSIDE}$. La classification a nécessite 57 secondes.

L'analyse de cette classification montre que :

- 2% des pavés de l'espace $[A_{C2}]$ ont été classés dans $\boldsymbol{A}^{INNER}$ et ne seront plus retraités (tous les points de ces pavés appartiennent à A_S) ;
- 90,9% des pavés de l'espace $[A_{C2}]$ ont été classés dans $\boldsymbol{A}^{OUTSIDE}$ et ne seront plus retraités (tous les points de ces pavés n'appartiennent pas à A_S) ;
- 7,1% des pavés de l'espace $[A_{C2}]$ ont été classés dans $\boldsymbol{A}^{BORDER}$ et on ne sait pas répondre aux questions :

- y a t'il au moins un point de ces pavés appartenant à A_S ?
- n'y a t'il aucun point de ces pavés appartenant à A_S ?

C'est pour cela que l'on parle de classe « Undetermined » et ces pavés devront donc être bissecté si l'on veut améliorer l'approximation.

On constatera donc l'efficacité de l'analyse car 92,9% des pavés de $[A_{C2}]$ auront été classés définitivement, c'est à dire que les points de ces pavés ont TOUS une position claire dans l'espace des ajustements. Il reste seulement 7,1% de pavés pour lesquels aucune réponse claire ne peut être donnée.

L'approximation de l'ensemble A_S est donc limité aux pavés classés dans $\boldsymbol{A}^{BORDER}$. Si l'on souhaite améliorer l'approximation, on devra diviser ces pavés afin de trouver de meilleures réponses au test de classification. On remarque donc que la division systématique génère trop de pavés identiques. Dans les exemples suivants nous modifierons la stratégie de division afin de générer le moins de pavés possible tout en conservant la qualité de l'approximation.

Le résultat de la classification a généré trois listes de pavés $\boldsymbol{L}^{INNER}$, $\boldsymbol{L}^{BORDER}$ et $\boldsymbol{L}^{OUTSIDE}$ qui donnent une description des sous-espaces respectifs. Nous représentons sur les figures suivantes la projection sur les trois plans des deux premières listes de pavés. Les pavés de la liste $\boldsymbol{L}^{INNER}$ sont représentés par des pavés verts pleins et les pavés de la liste $\boldsymbol{L}^{BORDER}$ sont encadrés de rouge.

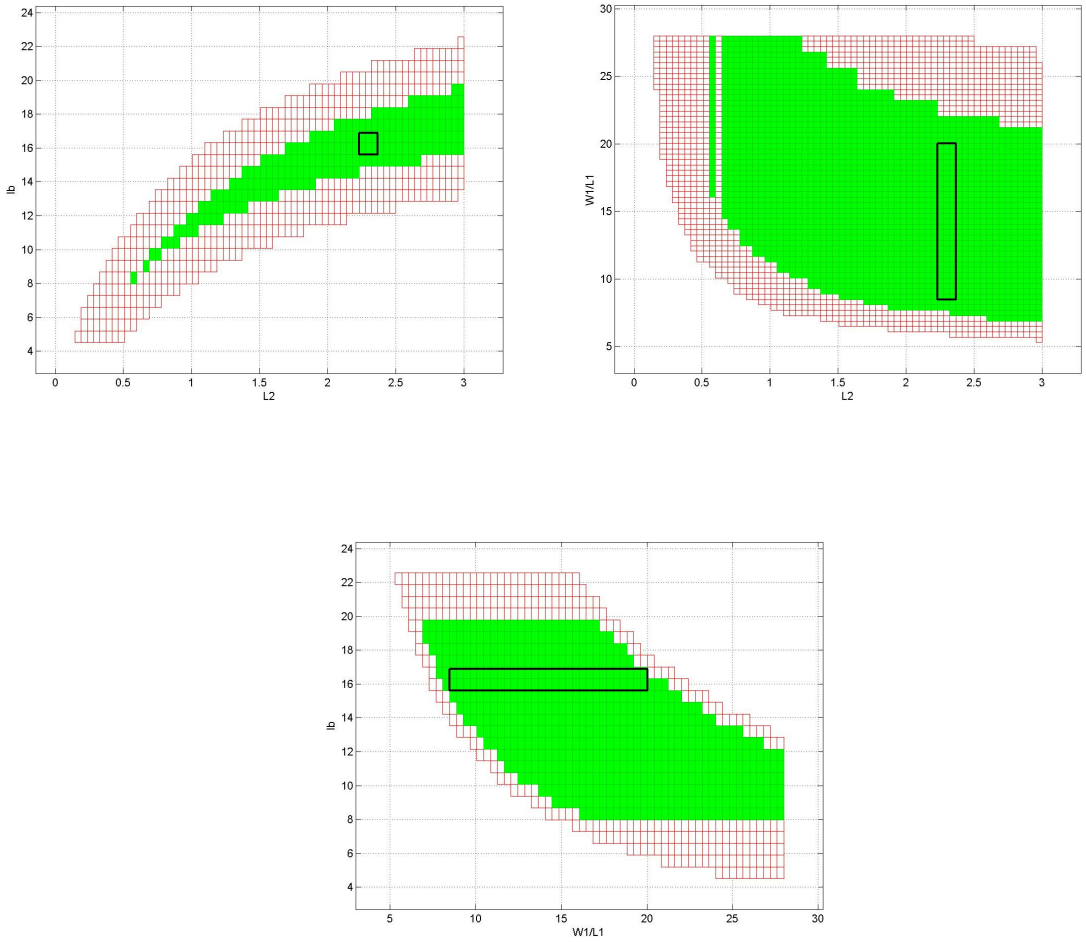


Figure 6-11 Représentation des projections des pavés intérieurs et de bords selon les plans L_2 - I_B, L_2 -$(W/L)_1$ et $(W/L)_1$- I_B

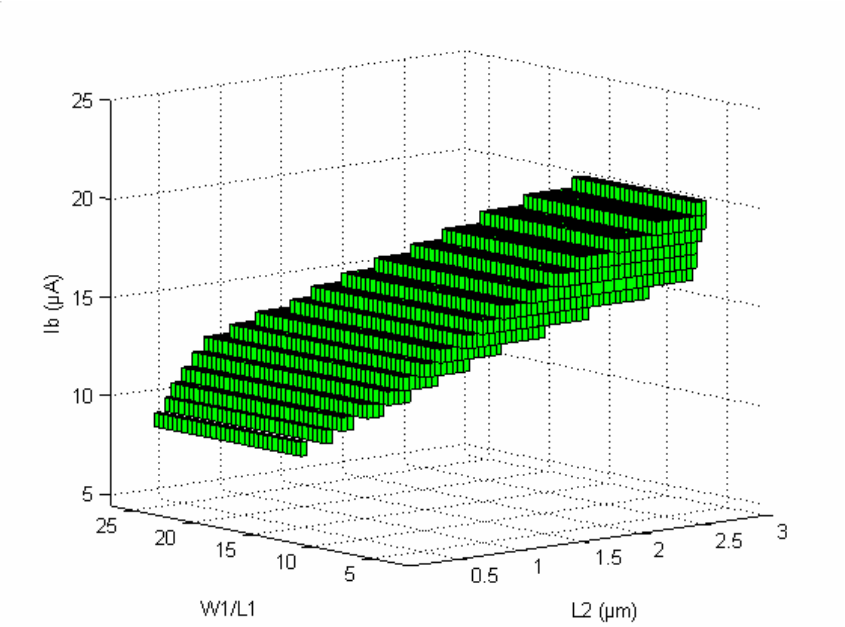


Figure 6-12 Représentation des projections des pavés intérieurs sur les trois dimensions

Sur la figure 6-2(a) (en haut à gauche : du plan I_B/L_2) on observera que les pavés Border créent une zone tampon entre les pavés OUTSIDE et les pavés INNER sauf sur le bord droit. Sur ce bord, l'espace faisable continu. Mais compte tenu de la restriction du domaine de recherche [*A*], les points à droite n'ont pas été traités. Par contre, à gauche, on est sûr que l'espace faisable est vide car une zone de Border sépare clairement les pavés INNER des Pavé OUTSIDE. Les mêmes remarques s'appliquent aux deux autres plans.

On observera la taille des pavés élémentaires. Leurs largeurs respectives suivant I_B, (W_1/L_1) et L_2 est de 0.8µA, 0.38 et 0.045µm. Ces valeurs peuvent être inférieures à la tolérance d'une mesure faite après production. De ce fait, diviser un tel pavé n'apportera rien d'utilisable en industrie. On devra donc veiller à ne pas diviser des pavés suivant une direction si celle-ci est inférieure à un seuil compatible avec les mesures de test ou de caractérisation. Cette remarquera sera utile pour définir le seuil d'arrêt des processus itératifs où l'on divisera jusqu'à atteindre ces seuils.

L'ensemble des pavés verts représente l'ensemble des points dont on est sûr qu'ils vérifient TOUTES les spécifications (autant composantes que fonctionnelles). A quoi peut donc servir d'avoir une infinité de points solutions si le dimensionnement ne doit retourner qu'un seul et unique point ? En fait, la connaissance d'un large « volume » (plutôt un large hyper-volume) permet de rechercher un pavé aussi large que possible dans chaque direction qui soit contenu dans ce volume. On choisira alors comme point nominal de dimensionnement le centre de ce « pavé inscrit ». Dans ce cas on est garanti que toute variation d'une variable autour du point nominal n'entraînera pas le non respect des contraintes du problème aussi longtemps que la variation reste inférieure à la demi longueur du pavé suivant sa dimension (sa direction). Par conséquent, le volume du pavé inscrit représente la robustesse de ce dimensionnement face aux dispersions de fabrication. Si la même analyse est faite sur plusieurs topologies comparativement, le plus grand pavé inscrit représentera alors la robustesse de ce dimensionnement pour cette topologie.

6.3 Seconde application à un circuit OTA et introduction d'une métrique de robustesse

Le second exemple sera découpé en deux parties. Dans la première partie nous voulons mettre en place un mécanisme d'extraction d'un pavé d'ajustements robuste aux variations du processus de fabrication. Dans ce but et pour pouvoir créer des images nous nous limiterons à deux variables indépendantes d'ajustement. Dans la deuxième partie, nous reprendrons cet exemple mais avec davantage de performances et d'ajustements. Nous montrerons alors que la démarche est possible sur un exemple plus proche du réel.

6.3.1 Redimensionnement de la paire différentielle de l'OTA

Le second support pour mettre œuvre notre méthode de redimensionnement est donc un OTA dont la fonction d'amplificateur est communément rencontrée dans les circuits intégrés analogiques et en particulier à l'interface dans les réseaux de neurones. Nous limiterons notre étude au seul redimensionnement de la paire différentielle de cet OTA en considérant l'expression de nouvelles spécifications comprenant le produit gain bande passante (GBW) et le gain différentiel (A_{Vd}). Le cas académique d'une phase classique de conception est présenté dans [Ant-04, Lak-94]. Nous reprendrons donc l'ensemble des données technologiques de ce travail et l'appliquerons à notre démarche.

Le schéma complet de l'OTA est donné sur la figure 6-4. La paire différentielle est constituée des transistors d'entrée M1 et M2 chargés par les transistors M3 et M4 placés en charges actives. La polarisation de la paire différentielle est assurée par le transistor M7 dont le courant est imposé par une source de courant externe I_{BIAS}. Les transistors M1 et M2 sont appariés c'est-à-dire ayant un rapport des géométries W/L identiques. Il en va de même pour les transistors M3 et M4.

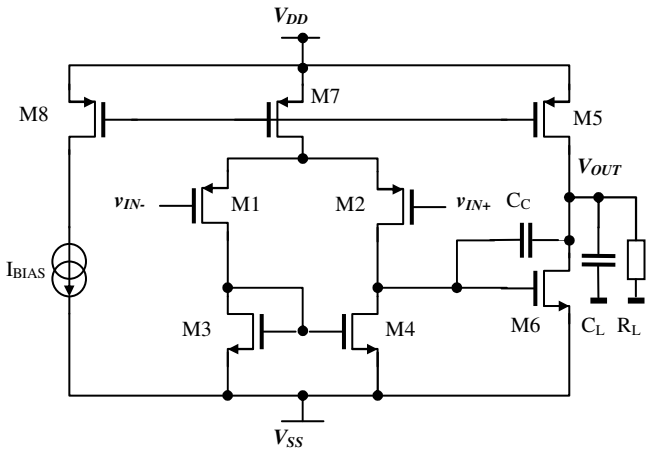


Figure 6-13 Schéma de l'OTA Miller

Les conditions de régime de fonctionnement des transistors en forte inversion impose une zone de travail des transistors au-delà du seuil $V_{GS} - V_{TH}$ avec $|V_{TH}| = 0.9V$. La tension d'alimentation V_{DD} minimale doit se situer dans une plage comprise entre 3V et 5V. Ces objectifs sont exprimés par des contraintes d'égalité et d'inégalité telles que :

$$3V \leq V_{DD} \leq 5V \quad (23),$$
$$V_{DSsat} = V_{GS} - V_{TH} \geq 0.1V \quad (24),$$
$$V_{DSmin} \geq V_{DSsat} \quad (25),$$
$$V_{DD} = V_{DSmin7} + V_{DSmin2} - V_{DSmin14} \quad (26),$$

$$I_{BIAS} = K_p \frac{W_2}{L_2} V_{DS\min}{}^2 \quad (27),$$

$$I_{BIAS} = \frac{K_p}{2} \frac{W_7}{L_7} V_{DS\min}{}^2 \quad (28),$$

$$I_{BIAS} = K_n \frac{W_4}{L_4} V_{DS\min}{}^2 \quad (29).$$

Les caractéristiques fonctionnelles sont le gain différentiel A_{Vd} et le produit gain bande passante GBW_{-3dB} donné à -3dB :

$$A_{Vd} = \frac{g_{m2}}{g_{ds2} + g_{ds4}} \bullet \frac{g_{m6}}{g_{ds5} + g_{ds6}} = \frac{2\sqrt{2\, K_N\, K_P \frac{W_2}{L_2} \frac{W_6}{L_6}}}{(\lambda_N + \lambda_P)^2\, I_{BIAS}} \quad (30)$$

$$GBW_{-3dB} = A_{Vd} \cdot f_{T(-3dB)} = A_{U1} \cdot A_{U2} \cdot \frac{g_{ds2} + g_{ds4}}{2\pi \cdot A_{U2} \cdot C_C} = \frac{1}{2\pi C_C} \sqrt{K_P I_{BIAS} \frac{W_2}{L_2}} \quad (31)$$

Enfin, les données technologiques sont les suivantes :

K_N = 95 µA/V², K_P = 21 µA/V², C_C = 1pF, $\lambda_n + \lambda_p$ = 29.7 m.V⁻¹, W_7/L_7 = 1, W_4/L_4 = 1 et W_6/L_6 = 23.

Les contraintes liées à la topologie du circuit et imposées par le concepteur [Lak-94] sont exprimées par :

- Le courant de polarisation est borné par : $0 \le I_{BIAS} \le 10^3$µA,
- Le rapport $(W/L)_2$ est borné par : $0.1 \le (W/L)_9 \le 100$,
- Le rapport $(W/L)_7$ est égal à : $(W/L)_7 = 1$,
- Le rapport $(W/L)_4$ est égal à : $(W/L)_4 = 1$,
- Le rapport $(W/L)_6$ est égal à : $(W/L)_6 = 23$,

Enfin, la symétrie du montage impose également des contraintes qui conduisent à vérifier l'égalité $W_1/L_1 = W_2/L_2$.

Le problème du dimensionnement peut alors se décomposer en deux CSP H_1 et H_2.

Les conditions à satisfaire par les composants du circuit sont formalisées dans H_1 :

$$f_{DB1}(a_{DB}) : \quad V_{DD} = V_{DSsat4} - V_{DSsat7} - V_{DSsat2}$$

$$\Leftrightarrow \quad V_{DD} = \sqrt{I_{BIAS}} \cdot \left(\sqrt{\frac{2}{Kp.(W_7 / L_7)}} + \sqrt{\frac{1}{Kp.(W_2 / L_2)}} + \sqrt{\frac{1}{Kn.(W_4 / L_4)}} \right)$$

$$\Leftrightarrow \quad V_{DD} = \sqrt{I_{BIAS}} \cdot \left(0.4112 + 0.2182 \sqrt{\frac{1}{W_2 / L_2}} \right) \quad (32)$$

$$f_{DB2}(a_{DB}) : I_{BIAS} = \frac{1}{2} Kp \frac{W_2}{L_2} V_{DSsat}{}^2 = 21 \frac{W_2}{L_2} V_{DSsat}{}^2, \quad (33)$$

$$f_{DB3}(a_{DB}) : I_{BIAS} = \frac{1}{2} Kp \frac{W_7}{L_7} {V_{DSsat}}^2 = 10.5 \frac{W_7}{L_7} {V_{DSsat}}^2 \quad (34)$$

$$f_{DB4}(a_{DB}) : I_{BIAS} = \frac{1}{2} Kn \frac{W_4}{L_4} {V_{DSsat}}^2 = 95 \frac{W_4}{L_4} {V_{DSsat}}^2 \quad (35)$$

avec les contraintes limitant la plage d'analyse :

$a_{DB} \in [A_{DB}] = [I_{BIAS}] \times [(W/L)_2] \times [V_{DSsat}] = [0.01, 50] \times [0.1, 100] \times [0.1, 20]$.

Le pavé des spécifications des composants est défini par :

$[S_{DB}] = [V_{DSsat}] \times [V_{DD}] = [0.1, 10^9] \times [3, 5]$

Le CSP H_2 porte sur les conditions à satisfaire par les caractéristiques fonctionnelles. Nous parlerons ici simplement de performances sans préciser qu'elles sont liées au fonctionnement du circuit. Les fonctions du CSP H_2 sont :

$$f_{P1}(a_P) : A_{Vd} = \frac{2\sqrt{2\, K_N\, K_P \frac{W_2}{L_2} \frac{W_6}{L_6}}}{(\lambda_N + \lambda_P)^2\, I_{BIAS}}, \quad (36)$$

$$f_{P2}(a_P) : GBW_{-3dB} = \frac{1}{2\pi C_C} \sqrt{K_P I_{BIAS} \frac{W_2}{L_2}} \quad (37)$$

avec les contraintes limitant la plage d'analyse :

$a_P \in [A_P] = [I_{BIAS}] \times [(W/L)_2] = [0.01, 50] \times [0.1, 100]$.

Les spécifications qui fixent les plages acceptables des caractéristiques fonctionnelles sont:

$[Sp] = [S\_A_{Vd}] \times [S\_GBW_{-3dB}] \div [80dB , 120dB] \times [3MHz , 10MHz]$.

Le CSP H global devra vérifier $H = H_1 \cap H_2$ et s'exprimera par :

$f_{DB}(a) \in [S_{DB}]$,
$f_P(a) \in [S_P]$,
$a \in [A] = [I_B] \times [(W/L)_1] \times [L_2] = [0.01, 50] \times [0.1, 100] \times [0.1, 3]$

6.3.2 Résolution partielle du CSP des composants : H_1

Les expressions (32) à (35) sont décomposées en primitives et exécutées récursivement. Le mécanisme de rétropropagation est détaillé dans le tableau 6-3.

Etape	Primitives (pOperators)	Fonction associée	Expression contractée
1	$[u_1] := [u_1] \cap sqrt[I_{BIAS}]$	$u_1 = sqrt(I_{BIAS})$	$f_{DB1}(a_{DB})$
	$[I_{BIAS}] := [I_{BIAS}] \cap ([u_1]*[u_1])$		
2	$[u_2] := [u_2] \cap sqrt[W_2/L_2]$	$u_2 = sqrt(W_2/L_2)$	
	$[W_2/L_2] := [W_2/L_2] \cap ([u_2]*[u_2])$		
3	$[u_3] := [u_3] \cap ([u_1]/[u_2])$	$u_3 = u_1/u_2$	
	$[u_1] := [u_1] \cap ([u_2]*[u_3])$		
	$[u_2] := [u_2] \cap ([u_1]/[u_3])$		
4	$[V_{DD}] := [V_{DD}] \cap (0.4112*[u_1]+0.2182*[u_3])$	$V_{DD} = 0.4112*u_1 + 0.2182*u_3$	
	$[u_1] := [u_1] \cap ([V_{DD}]/0.4112-0.2182/0.4112*[u_3])$		
	$[u_3] := [u_3] \cap ([V_{DD}]/0.2182-0.4112/0.2182*[u_1])$		
5	$[u_4] := [u_4] \cap ([V_{DSmin}]*[V_{DSmin}])$	$u_4 = V_{DSmin}^2$	$f_{DB2}(a_{DB})$
	$[V_{DSmin}] := [V_{DSmin}] \cap sqrt[u_4]$		
6	$[I_{BIAS}] := [I_{BIAS}] \cap (21*[W_2/L_2]*[u_4])$	$I_{BIAS} = 21*W_2/L_2*u_4$	
	$[u_4] := [u_4] \cap ((1/21)*[I_{BIAS}]/[W_2/L_2])$		
	$[W_2/L_2] := [W_2/L_2] \cap ((1/21)*[I_{BIAS}]/[u_4])$		
7	$[I_{BIAS}] := [I_{BIAS}] \cap (10.5*[V_{DSmin}]*[V_{DSmin}])$	$I_{BIAS} = 10.5*V_{DSmin}^2$	$f_{DB3}(a_{DB})$
	$[V_{DSmin}] := [V_{DSmin}] \cap ((1/10.5)*[I_{BIAS}]/[V_{DSmin}])$		
	$[V_{DSmin}] := [V_{DSmin}] \cap ((1/10.5)*[I_{BIAS}]/[V_{DSmin}])$		
8	$[I_{BIAS}] := [I_{BIAS}] \cap (95*[V_{DSmin}]*[V_{DSmin}])$	$I_{BIAS} = 95*V_{DSmin}^2$	$f_{DB4}(a_{DB})$
	$[V_{DSmin}] := [V_{DSmin}] \cap ((1/95)*[I_{BIAS}]/[V_{DSmin}])$		
	$[V_{DSmin}] := [V_{DSmin}] \cap ((1/95)*[I_{BIAS}]/[V_{DSmin}])$		

Tableau 6-8 Décomposition des expressions (32) à (35) en primitives (pOperators)

Le processus de contraction converge en 170ms après trois itérations et le sous espace $[A_{C1}]$ est donné par:

$$[A_{C1}] = C_{H1}([A],[S_{DB}])$$
$$[A_{C1}] = [I_{BIAS}] \times [(W/L)_2] \times [V_{DSsat}]$$
$$[A_{C1}] = [0.9, 50.0] \times [0.0,100.0] \times [0.1, 5.3].$$

6.3.3 Résolution partielle du CSP des fonctionnalités : H_2

Nous passons maintenant à la résolution du CSP H_2 qui reprend les caractéristiques fonctionnelles. Nous partons du pavé $[A_{C1}]$ (les solutions du CSP global H doivent également satisfaire H_1) et les points faisables au CSP global devrons appartenir à ce pavé.

Les équations de H_2 sont transformées en une suite d'opérateurs de contractions du type P-Opérateur et le pavé résultant sera $[A_{C2}]$.

Etapes	Primitives	Fonction associée	Expression contractée
1	$[u_1] := [u_1] \cap 2.04e4*sqrt[W_2/L_2]$	$u_1 = 2.04e4*sqrt(W_2/L_2)$	$f_{P1}(a_P)$
	$[W_2/L_2] := [W_2/L_2] \cap ([u_1]*[u_1]/2.04e4)$		
2	$[A_{V0}] := [A_{V0}] \cap ([u_1]/[I_{BIAS}])$	$A_{V0} = u_1/I_{BIAS}$	
	$[u_1] := [u_1] \cap ([I_{BIAS}]*[A_{V0}])$		
	$[I_{BIAS}] := [I_{BIAS}] \cap ([u_1]/[A_{V0}])$		
3	$[u_2] := [u_2] \cap ([W_2/L_2]*[I_{BIAS}])$	$u_2 = W_2/L_2*I_{BIAS}$	$f_{P2}(a_P)$
	$[I_{BIAS}] := [I_{BIAS}] \cap ([u_2]/[W_2/L_2])$		
	$[W_2/L_2] := [W_2/L_2] \cap ([u_2]/[I_{BIAS}])$		
4	$[GBW] := [GBW] \cap 7.2934e5*sqrt[u_2]$	$GBW = 7.2934e5 *sqrt(u_2)$	
	$[u_2] := [u_2] \cap ([GBW]*[GBW]/ 7.2934e5)$		

Tableau 6-9 Décomposition des expressions (36) à (37) en primitives (pOperators)

Après 3 itérations, l'algorithme converge en 60ms pour donner le pavé $[A_{C2}]$:

$[A_{C2}] = C_{H2}([A],[S_P])$
$[A_{C2}] = [I_{BIAS}] \times [(W/L)_2]$
$[A_{C2}] = [0.9, 20.4] \times [0.8,100.0].$

6.3.4 Exploration de l'espace de « conception » et description de l'espace faisable

La résolution du CSP *H* a conduit à une première approximation de l'espace faible A_S. Afin de réduire l'approximation, nous allons construire un sous-pavage de A_S et vérifier que l'image de ce sous-pavage appartienne à [*S*]. Il s'agira ensuite de proposer au concepteur le point nominal (I_{BIAS} et W_2/L_2) garantissant une marge de tolérance face aux dérives de fabrication du circuit. Cette opération sera présentée dans la prochaine section.
Avant cela, nous procédons à un pavage par division systématique du sous espace $[A_{C2}]$. La figure 6-5 donne les projections des pavés intérieurs et de bords satisfaisants les spécifications de gain différentiel et de produit gain bande passante. Le processus de classification a été exécuté en 4 secondes et à générer une liste de 119 pavés intérieurs et une liste de 201 pavés de bord. L'algorithme de découpage est stoppé lorsque la définition des pavés est inférieure à 1% de la plage initiale de chaque ajustement.

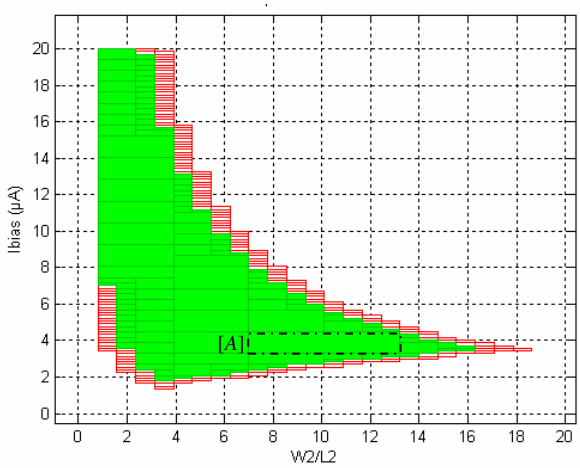


Figure 6-14 Représentation des projections des pavés intérieurs et de bord satisfaisants les spécifications de gain et de produit gain bande passante.

Partant d'un pavé du sous-espace A^{INNER}, il est possible de « dilater » un pavé qui constituera le pavé dont le centre sera doté de la plus grande marge de robustesse. Le pavé de départ à ce processus de dilatation peut être celui qui est placé en tête de la liste des pavés intérieurs $\boldsymbol{L}^{INNER}$. Il s'agira par conséquent du pavé ayant les plus grandes dimensions (le pavé $[A]_0$ sur la figure 6-5). Le processus de dilatation obéit à des règles et suit une procédure que nous allons développer.

6.3.5 Extraction d'un pavé robuste

L'exploration de l'espace des ajustements a permis de délimiter un sous espace vérifiant à la fois les contraintes imposées sur le circuit et les spécifications. Il s'agit à présent d'y dégager une région de confiance pour laquelle, le concepteur sera assuré de disposer d'un circuit respectant encore les objectifs visés vis-à-vis des dérives liées à la fabrication des circuits. La procédure consiste donc à construire une région qui confère aux performances le maximum de robustesse. Nous appellerons cette région le pavé de robustesse du sous espace des pavés intérieurs. Ce pavé de robustesse sera caractérisé par une largeur et une longueur constituant la métrique de notre démarche. Plus grandes seront les dimensions de ce pavé, plus grande sera alors la marge de robustesse.
L'idée générale de la procédure de robustesse est la suivante. Pour créer le pavé de robustesse, un pavé d'ajustement doit être retenu pour constituer le pavé de départ à notre procédure. Ce pavé sera pris en tête de la liste des pavés intérieurs fournis par la méthode de recouvrement. Il s'agira par conséquent du premier pavé classé dans la liste $\boldsymbol{L}^{INNER}$. Il suffit alors qu'un seul pavé soit classé dans la liste pour que la procédure de robustesse soit appliquée. Ceci souligne qu'il est possible de focaliser l'exploration de l'espace des ajustements à la recherche d'un seul pavé et ainsi d'en limiter le temps d'analyse. Le pavé retenu est ensuite dilaté d'un pas d'expansion que nous nommerons *d*. En fonction des performances qu'il renvoie, ce processus de dilatation est alors maintenu ou non. Tant que l'image du pavé de robustesse reste incluse dans les spécifications, le processus de dilatation est poursuivi. Dans le cas contraire l'expansion du pavé ayant entraîné un test d'inclusion négatif est stoppé.
Par exemple, le pavé $[A]_0$ est étendu dans toutes les directions d'un pas d=1/4 (volontairement pris large pour bien illustrer la méthode) pour devenir le pavé que nous nommerons $[A]_1$. Ce pavé est ensuite projeté dans l'espace des performances et son image est comparée au pavé de spécifications. La figure 6-6 illustre la projection du pavé $[A]_1$ et son inclusion dans le pavé des spécifications $[S]$. Le test d'inclusion étant positif, la seconde itération du processus est engagée et le pavé $[A]_1$ est à son tour dilaté du même pas pour devenir le pavé $[A]_2$. La figure 6-7 illustre ce cas de figure et le test d'inclusion se révèle là négatif ($[P]_2 \not\subset [S]$). L'extension du pavé est alors annulée et $[A]_1$ est donc désigné comme étant le pavé de robustesse que l'on notera $[A_{rob}]$.

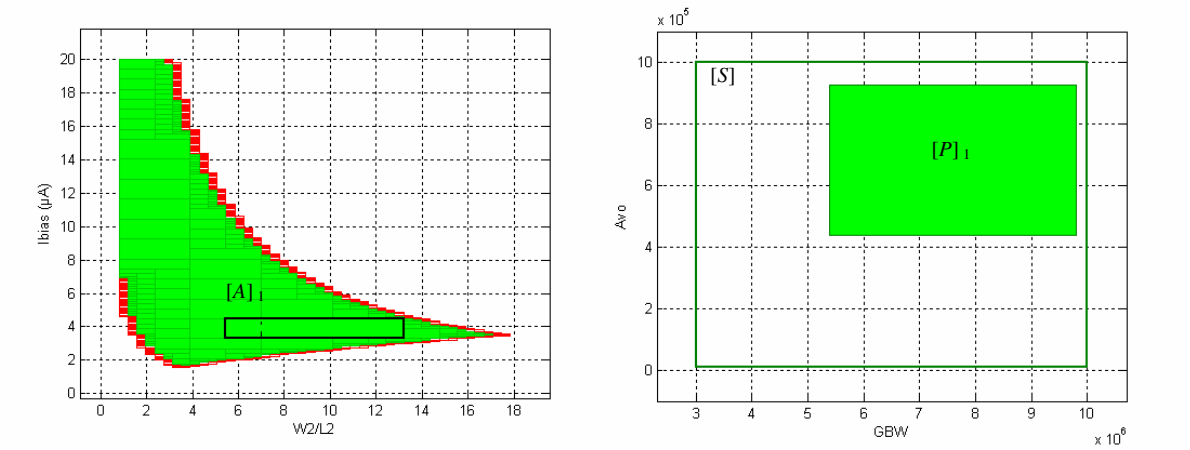


Figure 6-15 Projection du pavé intérieur $[A]_1$ dans l'espace des performances. L'image de $[A]_1$ est incluse dans les spécifications $[S]$.

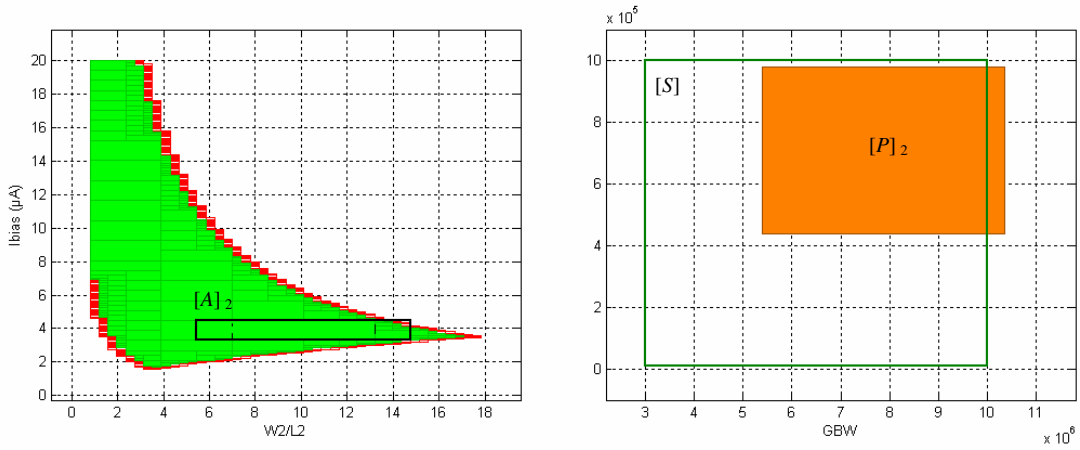


Figure 6-16 Projection du pavé intérieur $[A]_2$ dans l'espace des performances. L'image de $[A]_2$ n'est plus incluse dans les spécifications.

La stratégie employée à la dilation du pavé est bien entendu un facteur important dans la façon dont est construit le pavé final de robustesse dans le sous-espace solution. Une stratégie basée sur une extension symétrique des bords du pavé (l'ensemble des bords du pavé sont étendus simultanément) peut être restrictive vis-à-vis de certaines directions. La figure 6-6 est là pour le souligner tant le pavé final $[A]_1$ retenu pour être le pavé de robustesse n'est pas optimisé et une expansion des bords côtés bas et gauche peut encore être possible. En revanche, en optant pour une extension côté par côté, le pavé final de robustesse aboutit à une meilleure occupation de la surface du sous-espace solution. Le processus est détaillé ci-dessous pour une direction d'ajustement uniquement. On procède donc à l'extension d'un pas *d* pour une direction seulement. Le test d'inclusion entre performances et spécifications est alors effectué et de ce test dépendra donc de la poursuite ou non de l'extension dans la direction choisie. Dès lors qu'aucune direction ne permette une extension validant le test d'inclusion, le processus est donc stoppé et le pavé final retenu comme pavé de robustesse.
Procédure d'extension d'un pavé pour une direction d'ajustement *x* :

{
$[A]_0 = maxwidth(\mathbf{A}^{inner})$
For dimension x do
If $[\boldsymbol{f}]([A^x]_k = [x_k^- - d, x_k^+]) \subset [S]$ *Then*
$[A^x]_k = [x_k^- - d, x_k^+]$
Else
$[A_{rob}{}^x] \leftarrow [A_k{}^x]$
End If
If $[\boldsymbol{f}]([A^x]_k = [x_k^-, x_k^+ + d]) \subset [S]$ *Then*
$[A^x]_k = [x_k^-, x_k^+ + d]$
Else
$[A_{rob}{}^x] \leftarrow [A^x]_k$
End If
Exit loop when $[A_{rob}{}^x]$ *is fulfill for each x*
$k = k + 1$
End for
}

Par cette procédure, on réalise une nouvelle recherche du pavé de robustesse en prenant le pavé $[A]_0$ comme pavé de départ. La figure 6-8 illustre bien la nouvelle répartition du pavé final de robustesse obtenu après cinq itérations. Les dimensions du pavé sont données par les valeurs suivantes :

$$[A_{rob}] = [(W/L)_2] \times [I_{BIAS}] = [4.3, 8.5] \times [3.3, 4.5]$$

De ce pavé, il est alors intéressant d'en extraire le point central :

$$\boldsymbol{a_s} = ((W/L)_2=7.8, \quad I_{BIAS} =3.9\ \mu A)$$

Il paraît évident que de la finesse du pas choisi, le pavé final retenu sera tenu à une distance plus ou moins importante des bords de l'espace solution. La figure 6-8 illustre cet écart par les projections sur les axes des distances du coin du pavé aux pavés de bords ($\varepsilon_{W2/L2}$ and ε_{Ibias}) pour lesquelles les spécifications ne sont plus garanties.

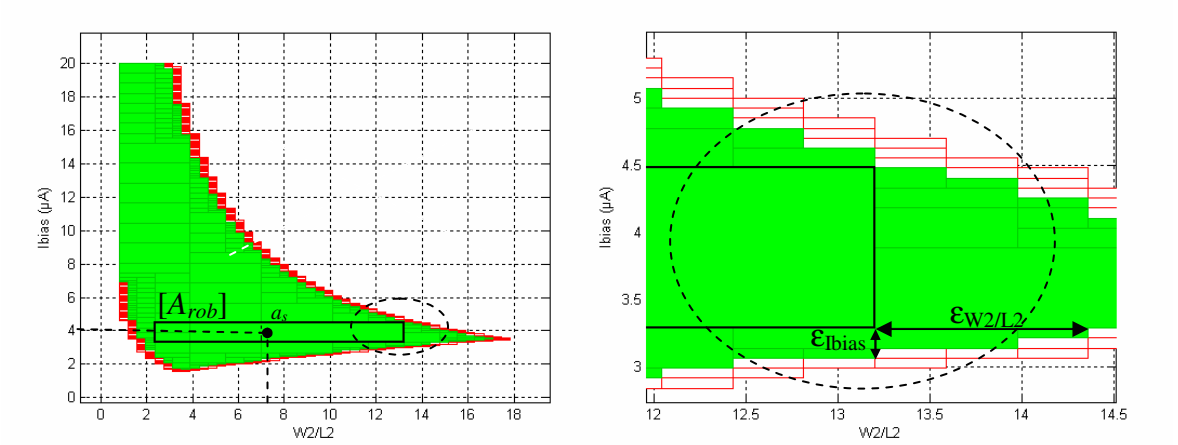


Figure 6-17 Représentation du pavé de robustesse $[A_{rob}]$ après cinq itérations avec son point d'ajustement central a_s (figure de gauche) pour un pas d=1/4. La distance du coin supérieur gauche du pavé de robustesse aux pavés de bords est caractérisée par la projection sur les axes des valeurs $\varepsilon_{W2/L2}$ et ε_{Ibias}

La figure 6-9 reprend l'espace solution et le tracé du pavé de robustesse lorsque le pas d'extension est fixé à d=1/10. La finesse du pas permet là encore d'améliorer la surface occupée par le pavé de robustesse et de se rapprocher de la bordure de l'espace solution.

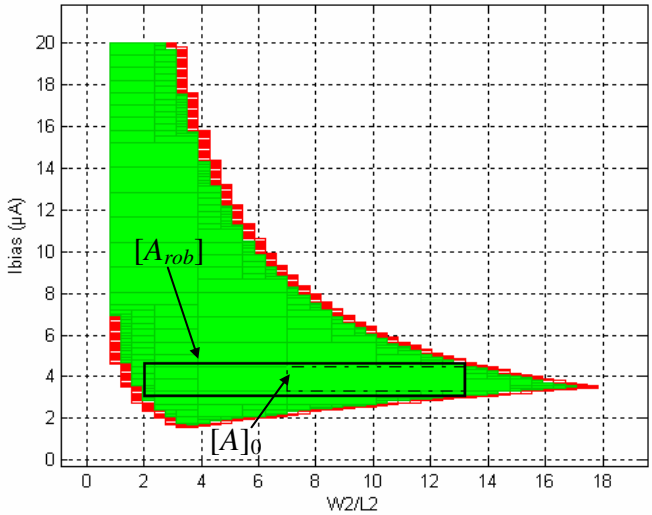


Figure 6-18 Représentation du pavé de robustesse $[A_{rob}]$ après onze itérations avec un pas d=1/10.

6.3.6 Redimensionnement complet de l'OTA

L'orientation pédagogique prise dans le paragraphe précédent nous a conduit à traiter notre problème de conception en limitant à deux le nombre de variables d'ajustements. Aussi, pour démontrer l'efficacité de la démarche, le redimensionnement de l'OTA est traité en considérant sept variables d'ajustements pour dix variables de performances.
La description de l'espace faisable sera traitée par division itérative. Le découpage d'un pavé en sous-pavés sera donc réalisé uniquement sur les pavés appartenant à la classe $\boldsymbol{A}^{BORDER}$ et un pavé classé dans $\boldsymbol{A}^{INNER}$ ne sera donc plus découpé. L'exploration de l'espace de conception sera conduite à partir de la stratégie associant la division itérative et la contraction de pavé. Cette stratégie a été présentée dans le chapitre 5-§5.7.

Quatre relations sont nécessaires pour formaliser l'expression des contraintes sur les régimes de fonctionnement des transistors. Le CSP H_1 traduit ces régimes de fonctionnement à partir des relations (38) à (41) :

$$f_{DB1}(a_{DB}) : \frac{g_m}{I} = \sqrt{K_P . \frac{W_1 / L_1}{I_{BIAS}}} \quad (38),$$

$$f_{DB2}(a_{DB}) : \frac{g_m}{I} = \sqrt{2.K_n . \frac{W_6 / L_6}{I_{D6}}} \quad (39),$$

$$f_{DB3}(a_{DB}) : \frac{g_m}{I} = \sqrt{2.K_P . \frac{W_7 / L_7}{I_{BIAS}}} \quad (40),$$

$$f_{DB4}(a_{DB}) : I_{TOT} = I_{BIAS} + I_{D6} \quad (41).$$

Les transistors de l'OTA fonctionnent dans leur ensemble dans un régime proche de la faible inversion. En conséquence, le rapport g_m/I_{DS} des transistors doit être situé au-dessus de 5 [Lak-94]. Nous limiterons enfin la consommation totale en courant à 30µA.

La symétrie du circuit impose par ailleurs de reprendre les mêmes dimensions géométriques pour les transistors appariés M1 avec M2 et M3 avec M4. Par conséquent :

$$W_1/L_1 = W_2/L_2 \quad (42);$$
$$W_3/L_3 = W_4/L_4 \quad (43);$$

Enfin, nous limiterons l'excursion des rapports W/L à 100.

La plage d'analyse des ajustements sera limitée à :

$$a_{DB} \in [A_{DB}] = [W_1/L_1]\times[W_3/L_3]\times[W_5/L_5]\times[W_6/L_6]\times[W_7/L_7]\times[I_{BIAS}]\times[I_{D6}]$$
$$= [0.01,100]\times[0.01,100]\times[0.01,100]\times[0.01,100]\times[0.01,100]\times[0.01,100]\mu A$$
$$\times[0.01,100]\mu A$$

Le pavé des spécifications des composants est défini par :

$$[S_{DB}] = [g_m/I_{DS}]\times[I_{TOT}] = [5\ ,\ 40]\times[0,\ 30]\mu A$$

Le jeu complet des équations de performances du circuit sont exprimées à l'aide des fonctions du CSP H_2:

$$f_{P1}(a_P) : GBW = \frac{\sqrt{\mathrm{Kn}.(\mathrm{W_1/L_1}).\mathrm{I_{BIAS}}}}{2\pi \mathrm{C_C}} \quad (44);$$

$$f_{P2}(a_P) : \mathrm{CMR\text{-}} = \mathrm{V_{SS}} + \mathrm{V_{Tn}} + \sqrt{\frac{\mathrm{I_{BIAS}}}{Kn.(W_1/L_1)}} + \sqrt{\frac{2.\mathrm{I_{BIAS}}}{Kn.(W_5/L_5)}} \quad (45);$$

$$f_{P3}(a_P) : \mathrm{CMR+} = \mathrm{V_{DD}} + \mathrm{V_{Tp}} + \mathrm{V_{Tn}} - \sqrt{\frac{\mathrm{I_{BIAS}}}{Kp.(W_3/L_3)}} \quad (46);$$

$$f_{P4}(a_P) : A_{v0} = \frac{1}{G_L'.g_{o24}} \sqrt{Kp\frac{W_1}{L_1} I_{BIAS}} \sqrt{2Kn\frac{W_6}{L_6} I_{D6}} \quad (47);$$

$$f_{P5}(a_P) : PM = 90 - \arctan\left(\frac{C_L'}{C_C} \sqrt{\frac{Kp.W_1/L_1.I_{BIAS}}{2.Kn.W_6/L_6.I_{D6}}} \right) \quad (48);$$

$$f_{P6}(a_P) : \mathrm{OUT\text{-}} = \mathrm{V_{SS}} + \sqrt{\frac{2.\mathrm{I_{D6}}}{Kn.(W_6/L_6)}} \quad (49);$$

$$f_{P7}(a_P) : \mathrm{OUT+} = \mathrm{V_{DD}} - \sqrt{\frac{2.\mathrm{I_{D6}}}{Kp.(W_5/L_5)}} \quad (50);$$

$$f_{P8}(a_P) : \mathrm{dv^2_{nife}} = 2.\left(\mathrm{dv^2_{nif1}} + \mathrm{dv^2_{nif3}}.\frac{\mathrm{Kn.W_3/L_3}}{\mathrm{Kp.W_1/L_1}} \right) \quad (51).$$

Avec pour limitations des variables d'ajustements :

$a_P \in [A_P] = [W_1/L_1]\times[W_3/L_3]\times[W_5/L_5]\times[W_6/L_6]\times[W_7/L_7]\times[I_{BIAS}]\times[I_{D6}]$
$= [0.01,100]\times[0.01,100]\times[0.01,100]\times[0.01,100]\times[0.01,100]\times[0.01,100]\mu A$
$\times[0.01,100]\mu A$

Les spécifications fonctionnelles sont les suivantes :

- la plage de tension de mode commun CMR s'étend de -0.5V à + 0.5V ;
- la dynamique de sortie de OUT- = -2.0V à OUT+ = +2.0V,
- le gain différentiel A_{Vo} doit être supérieur de 70dB,
- la marge de phase PM doit être supérieure à 60°,
- le produit gain bande passante GBW doit avoir un minimum de 1MHz,
- la contribution en bruit en 1/*f* des étages de l'OTA sera également prise en compte avec la réponse en bruit mesurée par la tension de bruit équivalente en entrée dv^2_{nife},
- les tensions d'alimentation sont symétriques et égales à ±2.5V.

Les plages portant sur les spécifications fonctionnelles sont alors :

$[Sp] = [S\_GBW]\times[S\_CMR\text{-}]\times[S\_CMR+]\times[S\_A_{Vo(dB)}]\times[S\_PM]$
$\times[S\_OUT\text{-}]\times[S\_OUT+]\times[S\_dv^2_{nife}]$
$= [1\ MHz,10MHz]\times[-2.5V,-0.5V]\times[+0.5V,+2.5V]]\times[70dB,120dB]\times[60°,90°]$
$\times[-2.5V,-2.0V]\times[+2.0V,+2.5V]\times[0,10^{-9}V^2_{RMS}/Hz]$

Enfin, les données technologiques sont les suivantes :

$$K_n = 95\ \mu A/V^2,\ K_p = 21\ \mu A/V^2,\ C_C = 1pF,\ C_L' = 10.4pF,\ V_{THp} = -0.66V,$$
$$V_{THn} = 0.58V,\ G_L' = 11.8\mu S,\ g_{o24} = 0.03\mu S.$$

Etant donné que seules les contributions de bruit des transistors M1 et M3 participent le plus largement à la contribution totale du circuit [Lak-94], nous utiliserons donc pour M1 la contribution $dv^2_{nif1} = 1.13\mu V_{RMS}/Hz^{-1/2}$ et pour M3 $dv^2_{nif3} = 10.56\mu V_{RMS}/Hz^{-1/2}$.

6.3.7 Application de notre stratégie d'exploration de l'espace de conception

La stratégie d'exploration de l'espace des ajustements adoptée pour le dimensionnement de l'OTA a été présentée dans le chapitre 5-§5.7. Cette stratégie repose sur un processus itératif faisant intervenir une opération de division, de classification et de contraction. L'exécution de ce processus est paramétrable par son utilisateur et un certain nombre de possibilités s'offre alors à lui quant au choix de :

Définir la méthode de contraction à partir des contraintes sur les ajustements seules, la méthode de contraction à partir des contraintes sur les performances seules ou les deux à la fois.
Contracter initialement l'espace des ajustements c'est-à-dire d'appliquer la méthode de contraction avant de procéder à une analyse fine de cet espace.
De choisir entre la technique de découpage par bissection ou par trisection. Pour chacune de ces techniques le choix du découpage des ajustements à traiter est défini soit selon la dimension la plus importante (dimension la plus sensible) auquel cas cette dimension sera désignée pour être découpée, soit de manière indifférenciée et toutes les dimensions sont alors découpées.
De réduire la taille des listes à visualiser en procédant à une réunion des pavés dont l'une des composantes est identique.

La stratégie d'exploration choisie consiste à résoudre le CSP global $H = H_1 \cap H_2$ à l'aide des opérateurs de contraction. Par récurrence, l'espace solution est ensuite approché en alternant les méthodes de contraction par P-opérateurs et d'inversion ensembliste. L'inversion ensembliste s'appuie sur une division itérative de l'espace par trisection.

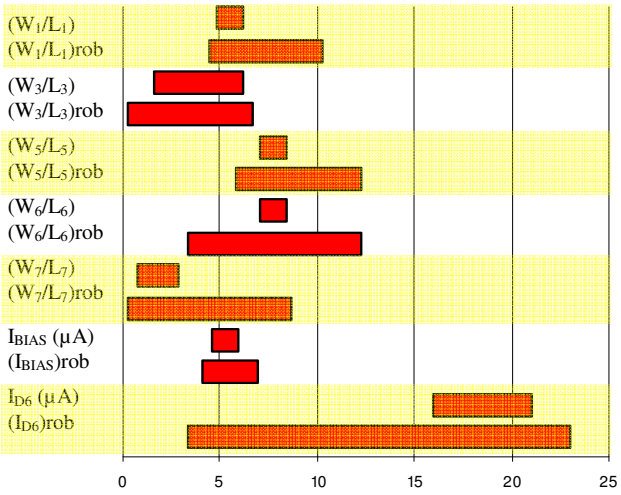
L'exploration est effectuée en 21 secondes et retourne le pavé $[A]_9$ classé dans la liste des pavés intérieurs après 9 itérations. A ce pavé est ensuite appliqué le processus de robustesse et une dilatation conduit aux résultats exposés dans le tableau 6-5.

Ajustements	**Intervalles du pavé $[A]_9$**	**Intervalles du pavé $[A_{rob}]$**
W_1/L_1	[4.8, 6.2]	[4.5, 10.3]
W_3/L_3	[1.6, 6.2]	[0.2, 6.7]
W_5/L_5	[7.0, 8.4]	[5.8, 12.2]
W_6/L_6	[7.0, 8.4]	[3.4, 12.2]
W_7/L_7	[0.8, 2.9]	[0.3, 8.7]
I_{BIAS} (µA)	[4.6, 6.0]	[4.1, 6.9]
I_{D6} (µA)	[16.0, 21.0]	[3.4, 23.0]

Tableau 6-10 Résultats des intervalles pour les pavés d'intérieur $[A_0]$ et de robustesse $[A_{rob}]$.

Les intervalles du tableau 6-5 sont représentés ci-dessous par des barres horizontales juxtaposées pour faciliter la comparaison entre les intervalles calculés avant et après la procédure de robustesse. On constate qu'après dilation, les intervalles des ajustements W_1/L_1, W_5/L_5, W_6/L_6, W_7/L_7 et I_{D6} ont subi une multiplication de leur largeur d'un facteur 4. L'impact sur le

positionnement du point nominal est donc fortement lié au facteur de dilation mais aussi à la façon dont s'est dilaté l'intervalle : une dilation symétrique de part et d'autre de l'intervalle « conservera » l'emplacement du point nominal (cas de l'ajustement W_3/L_3 ou W_6/L_6) alors qu'une dilation asymétrique de l'intervalle entraînera un déplacement plus marqué du point nominal (cas de l'ajustement W_1/L_1, W_7/L_7 ou I_{D6}). Dans ce dernier cas, la procédure de robustesse est déterminante et conduit à un « repositionnement » très clair du point nominal.



De ce pavé de robustesse, on en extrait donc le point nominal correspondant au centre géométrique de $[A_{rob}]$:

$\boldsymbol{a_s}$ = (W_1/L_1 = 7.4, W_3/L_3 = 3.4, W_5/L_5 = 8.9, W_6/L_6 = 7.8, W_7/L_7 = 4.5, I_{BIAS} = 5.5µA, I_{D6} = 13.2µA)

Nous allons maintenant nous assurer que les intervalles du pavé de robustesse garantissent toujours une parfaite adéquation entre performances et spécifications. On effectue un tirage aléatoire de 1000 points uniformément sur l'espace des ajustements en prenant comme sous-espace cible le pavé de robustesse $[A_{rob}]$. Un calcul des performances correspondantes permet ensuite de faire la comparaison avec le pavé des spécifications. Le tableau 6-6 reprend le comparatif et l'ensemble des points de performances est contenu dans l'intervalle des spécifications.

Variables de performances	**Intervalles des spécifications $[S_{DB}]$ et $[S_P]$**	**Intervalles des performances $[P_{Arob}]$**
g_m/I_{DS}	[5 , 40]	[12.9 , 25.2]
I_{TOT} (µA)	[0, 30]	[7.6, 29.9]
GBW (MHz)	[1 , 10]	[5.5 , 10.7]
CMR- (V)	[-2.5 , -0.5]	[-2.49 , -1.90]
CMR+ (V)	[+0.5 , +2.5]	[+0.52 , +1.55]
$A_{Vo(dB)}$ (dB)	[70 , 120]	[70.1 , 89.7]
PM (deg)	[60 , 90]	[88.5 , 89.2]
OUT- (V)	[-2.5 , -2.0]	[-2.42 , -2.12]
OUT+ (V)	[+2.0 , +2.5]	[+2.00 , +2.37]
dv^2_{nife} (V^2_{RMS}/Hz)	[0 , 10^{-9}]	[12.9 , 979.5].10^{-12}

Tableau 6-11 Comparatif des résultats entre l'image des points pris dans le pavé de robustesse $[P_{Arob}]$ et le pavé des spécifications.

La figure 6-10 donne une illustration du tirage aléatoire pris sur le pavé de robustesse [A_{rob}] où seuls les ajustements W_5/L_5 et W_6/L_6 ont été représentés. Les points du tirage sont ensuite projetés dans l'espace des performances en se limitant à la représentation de la dynamique de sortie de l'OTA (OUT- et OUT+). Cette projection conduit à la formation d'un amas de points situé à l'intérieur du pavé des spécifications [S]. La projection dans l'espace des performances du point nominal $\boldsymbol{a_n}$ extrait du pavé de robustesse fournit un point $\boldsymbol{p_n}$:

$\boldsymbol{p_n}$ = (OUT- = -2.309V, OUT+ = 2.197V)

Ce point $\boldsymbol{p_n}$ est situé à une distance $\boldsymbol{\Delta out}$ des bords du pavé des spécifications :

$\boldsymbol{\Delta out}$ = ($[\Delta out\text{-}]^-$ = 0.191V, $[\Delta out\text{+}]^-$ = 0.197V, $[\Delta out\text{-}]^+$ = 0.309V, $[\Delta out\text{+}]^+$ = 0.303V)

Cette distance permet alors d'évaluer la marge de robustesse des performances du point nominal par rapport aux spécifications.

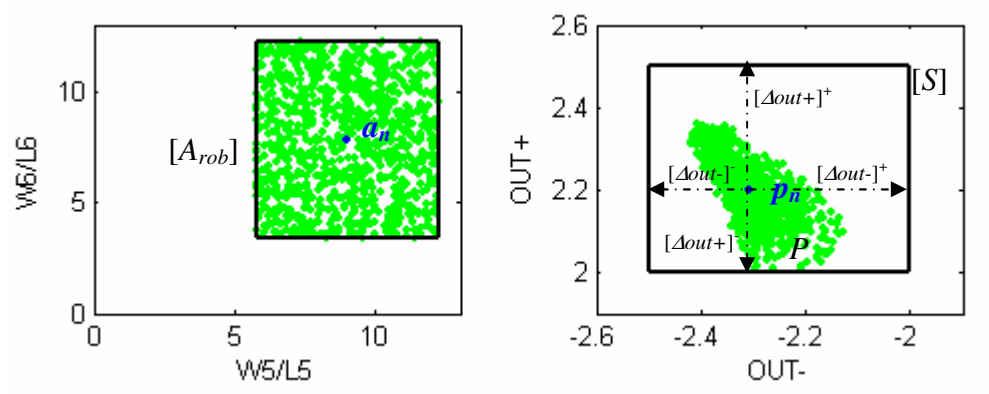


Figure 6-19 Représentation des points images (dynamique de sortie OUT- et OUT+) pris dans l'intervalle de robustesse [A_{rob}] (seuls les ajustements W_5/L_5 et W_6/L_6 ont été représentés).

La figure 6-11 illustre par un histogramme la projection du tirage uniforme pris à partir du pavé de robustesse.

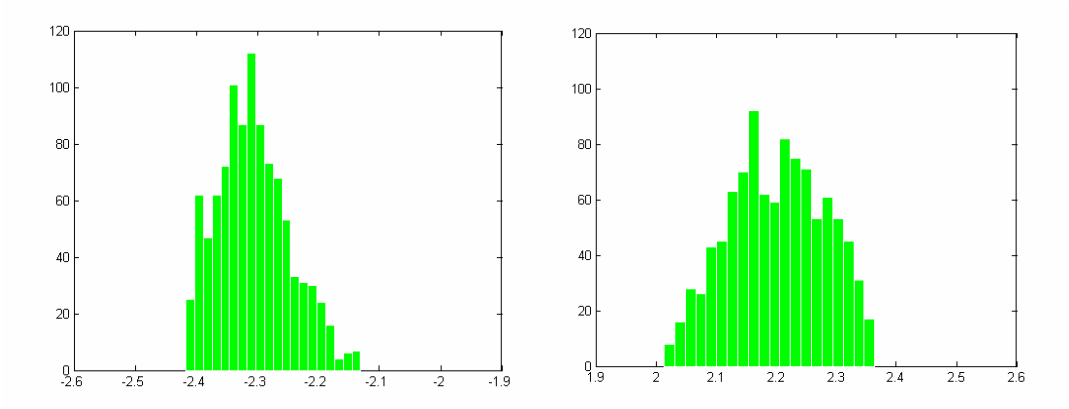


Figure 6-20 Histogramme de la projection des performances OUT- (à gauche) et OUT+ (à droite) pour un tirage de 1000 points.

A partir des données de l'histogramme, on calcule la valeur moyenne de la distribution des grandeurs de performances OUT- et OUT+ :

<OUT-> = -2.307V
<OUT+> = 2.200V

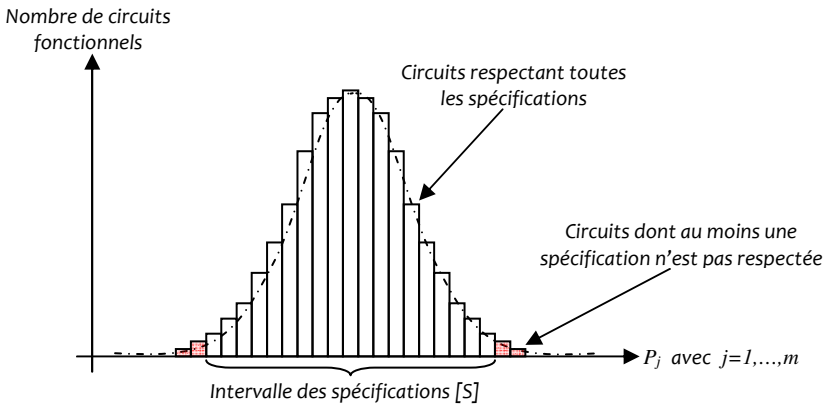
On observe que l'on retrouve des valeurs identiques à celles calculées avec le point nominal $\boldsymbol{p_n}$. La projection dans l'espace des performances du point nominal $\boldsymbol{a_n}$ qui correspond au centre du pavé de robustesse produit un point $\boldsymbol{p_n}$ qui est la moyenne de la distribution de l'ensemble des points de performances.

6.4 Mesure de la robustesse dans la conception des circuits intégrés

La fabrication d'un circuit intégré est affectée de phénomènes de variabilités dont l'impact sur ses performances peut conduire à un circuit non valide au regard des spécifications et finalement à son rejet. Chaque fabricant de circuits calcule le taux de rejet par une analyse de rendement de production qu'il définit par le rapport de la part des circuits satisfaisant toutes les spécifications sur l'ensemble des circuits fonctionnels.
Le rendement correspond à la probabilité qu'un circuit respecte toutes les spécifications sur l'ensemble de ses performances. Il peut donc s'exprimer à partir du domaine des spécifications **S** :

$$\eta = \int_{[S]} f_{P_m}(P_1,...,P_m).dP_1...dP_m$$

où f_{Pm} est la densité de probabilité conjointe des performances P_1 à P_m. L'analyse du rendement de production vise à construire la fonction de densité f_{Pm} qui peut être approchée à l'aide d'un histogramme comptabilisant le nombre de circuits fonctionnels ayant passés le test d'acceptabilité vis-à-vis des spécifications :

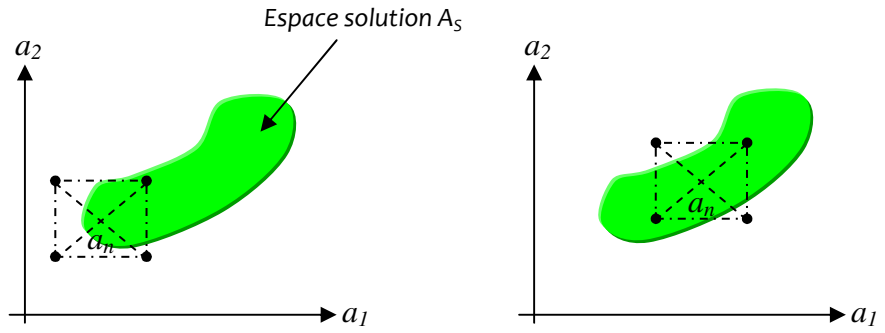


Afin de maximiser le rendement de production, une analyse de rendement est alors nécessaire et doit être réalisée au plus tôt dans le cycle de conception. L'objectif visé consiste à reproduire a priori l'histogramme qui est une estimation de la distribution des performances. L'histogramme recouvre la loi de probabilité elle-même caractérisée par une valeur moyenne et un écart-type σ. L'objectif du concepteur est alors de rendre la conception d'un circuit robuste aux variations des paramètres du processus de fabrication. Le concepteur calcule pour cela le point nominal de fonctionnement autour duquel un intervalle peut être maximisé dans le respect des spécifications. On définit cela comme l'analyse de robustesse ou test de robustesse du circuit. Ce voisinage est en pratique fixé à [-3σ, +3σ] autour de la moyenne dans l'hypothèse d'une distribution Gaussienne.

Néanmoins, construire une loi de probabilité pour l'ensemble des performances suppose a priori de connaître les fonctions de densités de probabilités de chaque variable d'ajustement. Ces fonctions de densité ne sont pas connues avec exactitude et évoluent durant le processus de fabrication du circuit. De plus, pouvoir expliciter la loi de probabilité des performances nécessite de connaître les lois de covariance entre variables, ce qui n'est pas le cas.

La solution couramment utilisée par les industriels est alors de mettre en œuvre, à l'aide des outils de simulation, des mesures statistiques afin de caractériser la distribution des performances. Un échantillon de points de l'espace des ajustements est projeté dans l'espace des performances et un test d'acceptabilité est ensuite effectué. La moyenne de ce test permet alors de qualifier l'espace analysé en terme de robustesse. L'analyse Monte-carlo par exemple fournit des résultats précis mais qui nécessitent un effort calculatoire et un temps d'analyse importants. Une analyse moins coûteuse en terme de temps est l'analyse du « pire cas ». Avec cette analyse, le circuit est simulé à partir de modèles construits en prenant les valeurs extrêmes des paramètres technologiques des composants (tension de polarisation, dopage, dimensions géométrique des transistors, …). L'espace contenu à l'intérieur des coins correspond à l'espace de tolérance fixé par la technologie. Les performances obtenues sont donc les performances extrêmes qui sont ensuite comparées aux bornes de l'intervalle des spécifications. Ce type d'analyse présente un temps d'analyse faible mais peut conduire à une prédiction des performances surévaluée.

L'idée de compromis entre les deux dernières approches (Monte-Carlo et Analyse du pire cas) conduit à ce que l'on appelle le « Design Centering » ou « dimensionnement par centrage ». Il s'agit alors d'effectuer une première approche de l'espace solution des ajustements qui maximise le test d'acceptance par une analyse du pire cas. Partant du point nominal de fonctionnement a_n, un processus d'optimisation va chercher à inscrire le maximum de points extrêmes ou « coins » à l'intérieur de l'espace solution. La figure 6-12 illustre deux situations qui conduisent à inclure progressivement les coins délimitant l'espace de tolérance à l'intérieur de l'espace solution. Un point nominal proche du bord de l'espace solution (figure de gauche) présente une marge de robustesse faible et donc un risque de détérioration du rendement de production. A l'inverse (figure de droite), un point nominal situé au milieu de l'espace solution présente une marge de robustesse plus importante et donc réduit le risque de détérioration du rendement de production.



Point nominal a_n proche du bord de A_S

Point nominal a_n situé au centre de A_S

Figure 6-12 : recherche du volume maximum de l'espace de tolérance à l'intérieur de l'espace solution A_S.

Optimiser le rendement revient à maximiser l'intersection du volume de l'espace de tolérance délimité par les coins avec le volume de A_S c'est à dire rechercher le plus grand volume de l'espace de tolérance inclus dans A_S. Ceci revient à déplacer le centre de gravité (barycentre) de

l'espace de tolérance afin de maximiser le test de robustesse. L'analyse du pire cas est ensuite complétée par une analyse de type Monte-Carlo qui donne une estimation du rendement de production par une procédure de validation : les points contenus dans l'espace de tolérance font-ils tous correspondre des performances contenus dans les spécifications ?
Cette procédure de placement du centre de gravité accompagnée de l'échantillonnage de l'espace de tolérance sera pertinente si l'on place tous les coins dans l'espace solution c'est-à-dire si l'on vise un test d'acceptance de 100%. Par conséquent, les techniques de Design Centering sont davantage des techniques d'optimisation que d'analyse de rendement.

Dans la littérature deux grandes approches sont généralement choisies pour appliquer une procédure de centrage : l'approche statistique (Monte-carlo) ou l'approche géométrique.

[Gra-92] rétro propage les bords des performances (des spécifications) vers les ajustements pour donner les bords frontières de la région solution et en déduire les distances maximales garantissant le respect des spécifications. Il en extrait un point « central » qui est en fait le « centre » de la région de tolérance qui elle est ellipsoïdale. [Sap-93] se base sur l'hypothèse d'un espace faisable convexe. Il approche l'espace faisable par une recherche de la meilleure ellipse qui s'inscrit dans son polytope. Le centre de cette ellipse sera donc désigné comme le point central. Le travail est basé sur un processus d'optimisation convexe (géométrique) et sur un jeu d'équations des performances. [Sac-94] cherche également à approximer l'espace des ajustements par un polytope formé par l'intersection d'un ensemble de demi-plans. Le placement du point central est effectué en inscrivant la plus grande ellipse dans le polytope par la définition d'une matrice Hessienne. Cette approche se montre efficace en considérant des problèmes convexes uniquement. [Ant-94] effectue une analyse du pire cas pour mesurer l'écart entre les performances et spécifications. Cette analyse permet donc d'évaluer, pour chaque performance prise individuellement, la tolérance maximale d'un paramètre. Une zone en courbes est tracée et délimite la région solution des paramètres de fonctionnement. Ces courbes sont croisées avec des ellipsoïdes dont le centre est un point de la région solution (correspondant aux paramètres statistiques) et qui caractérisent des contours de la fonction de densité de probabilité (courbes d'équiprobabilité) du processus de fabrication. Ce centre est désigné comme étant le point nominal de fonctionnement à partir duquel l'analyse du pire cas est construite. La condition de tangence entre les courbes de la région solution et le contour de la fonction de densité permettent alors d'en extraire un polytope. [Xin-05] cherche à maximiser les performances du circuit à l'intérieur d'une forme ellipsoïdale. La technique de d'approximation utilise les fonctions posynomiales. Le volume résultant de l'ellipse est utilisé comme critère de sélection : le plus grand volume implique la plus grande probabilité pour réussir une implémentation du circuit. [Xu-05] s'intéresse à l'introduction de l'analyse de la robustesse dans les circuits intégrés par une approche « Worst case ». Une optimisation force alors les coins du pavé des performances dans les spécifications.

La synthèse de ces travaux démontre l'intérêt de l'approche géométrique dans le but d'extraire un sous-espace faisable à partir duquel un centre de gravité sera calculé et désigné comme point nominal de fonctionnement. Plus le sous-espace faisable sera grand et plus le circuit sera robuste et offrira un rendement de production élevé. Parmi ces travaux, l'extraction du sous-espace faisable est encore conduite avec des analyses ponctuelles. Une exploration de l'espace des ajustements est alors typiquement menée par un échantillonnage aléatoire. La recherche du circuit le plus robuste est ensuite obtenue par un algorithme d'optimisation qui calcule un seul point solution sans retourner la moindre information sur la propriété des points de son voisinage.

Avec l'analyse par intervalles, le traitement des sous-espaces par leur union ou leur intersection retourne rapidement des informations utiles (volume espace faisable) qui permettront de

maximiser l'espace de tolérance dans l'espace faisable. L'espace de tolérance sera un pavé simple à calculer mais dont la représentation peut être plus pessimiste qu'une représentation ellipsoïdale conduisant finalement à sous-estimer le rendement. La représentation par pavé revient à une analyse par pire-cas dont le problème se ramène à centrer l'espace de tolérance au-dessus de l'espace faisable A_S afin de maximiser le volume résultant de l'intersection. Notre approche reprend finalement l'idée d'une approche par centrage : maximiser l'espace de tolérance dans l'espace faisable et en extraire le centre de gravité.

6.5 Conclusion

Dans ce dernier chapitre, nous avons mis en application la stratégie d'exploration de l'espace de conception et la description de l'espace faisable à l'aide des approches par contraction et d'encadrement. Plusieurs exemples ont été choisis et permettent de poser les premiers jalons d'une démarche de dimensionnement d'un circuit.
L'étage de sortie d'un amplificateur a d'abord été dimensionné en fonction de contraintes de fonctionnement et en spécifiant un ensemble d'objectifs de performances. La simplicité du problème et le nombre d'ajustements limité ont permis de dégager très rapidement un sous-espace solution dans lequel un point nominal de fonctionnement a pu être sélectionné. Ce choix est crucial et permet d'attribuer au circuit un réglage qui donne la plus grande marge de fonctionnement à l'encontre des aléas liés au processus de fabrication. Il est alors possible d'inscrire un volume à l'intérieur du sous espace solution permettant de définir et de comparer des facteurs de qualité (par rapport à un gain, une bande-passante, …) entre différentes topologies. Le calcul de ce volume a été ensuite appliqué sur un amplificateur opérationnel à transconductance à partir d'une exploration de l'espace des ajustements basée sur l'exécution conjointe de la contraction de pavé et de la division itérative. La valeur nominale des composants du circuit a été extraite d'un pavé de robustesse. Ce calcul, qui conduit à définir une marge de robustesse, ouvre la porte à de nouvelles méthodes permettant d'attribuer un facteur de qualité à une topologie de circuit donnée. Ce travail associe finalement le dimensionnement d'un circuit et la sélection de topologies. Cet aspect de la sélection de topologies sera abordé plus en détail dans le dernier chapitre.

Chapitre 7 – conclusion générale

Ce chapitre propose de synthétiser la démarche de conception qui fut la notre à travers ce travail de recherche. Cette synthèse nous permettra ensuite de resituer notre action au sein du cadre global du développement de circuits intégrés mixtes. Nous verrons enfin comment ces étapes s'inscrivent dans le concept du prototypage virtuel fonctionnel pour répondre au risque industriel d'une conception d'un circuit mixte non aboutie.

Les problématiques traitées

Dans cette thèse, notre travail s'est focalisé sur deux problématiques:

- le traitement massivement parallèle de données ;
- l'efficacité de conception des circuits analogiques.

Pour répondre à la première problématique, nous nous sommes tournés vers les réseaux de neurones qui suscitent depuis plusieurs années un engouement grandissant auprès de la communauté de la recherche. Les nouvelles technologies conduisent à une augmentation du volume de données à traiter. Les limites d'un traitement séquentiel de ces données intensifient le recours aux architectures parallèles pour augmenter la puissance de calculs. Aussi, on assiste à une effervescence autour des réseaux de neurones qui tirent les avantages des architectures parallèles par l'interconnexion massive de centaines voir de milliers d'unités de calcul. Le fort parallélisme des unités de calculs d'un réseau de neurones, la minimisation du rapport énergie/capacité de traitement, la capacité à s'adapter à des situations imprévues ou à identifier des processus physiques non formalisables, justifient entre autre l'intérêt croissant qui leur est accordé. C'est sur ce modèle de traitement de l'information que nous avons développé une application illustrative dédiée à la reconnaissance de caractères.

Concevoir une application à réseaux de neurones analogiques nécessite de poser une démarche rigoureuse de conception. Cette action vise à répondre à la seconde problématique de l'efficacité de conception.

Notre démarche se trouve au centre d'un fort besoin industriel. Elle s'inscrit dans la problématique d'une méthodologie globale pour concevoir des systèmes complexes et multidisciplinaires sous la contrainte du '*first time succes*'. En ce sens, cette méthode doit permettre d'obtenir le plus rapidement possible une solution de conception optimale en temps au regard des spécifications. Elle vise à améliorer les outils de conception en reprenant toutes les étapes du flot de conception partant de l'idée d'un produit jusqu'à sa réalisation.

C'est dans ce contexte pluri-objectifs que s'inscrit notre démarche. Notre travail associe finalement la conception de réseau de neurones et les méthodes rendant sa conception efficace.

Synthèse du travail

La première étape de notre démarche fut de procéder à une analyse conceptuelle de l'application neuronale dans son projet applicatif (figure 1). Cette analyse a permis de valider le choix de l'architecture '*feedforward*' du réseau en mesurant sous Matlab l'influence de l'algorithme d'apprentissage '*weight perturbation*' lorsque le mode de perturbation des poids synaptiques est parallèle.

Lors de la seconde étape, l'application a été décomposée en un jeu de blocs élémentaires interconnectés. Pour vérifier le fonctionnement de l'application dans sa globalité, les blocs ont été développés à différents niveaux de description. Le passage entre ces niveaux de descriptions

est aujourd'hui facilité par l'utilisation du langage de description normalisé IEEE VHDL-AMS 1076.1. Une telle méthodologie permet ainsi de vérifier et d'affiner les différentes descriptions du système tout au long du processus de conception dans un environnement unifié de simulation. Les modèles comportementaux du réseau et de son algorithme d'apprentissage ont été élaborés en VHDL-AMS en s'assurant du « portage » du modèle développé à partir de MATLAB. Une analyse comparative a permis de vérifier l'équivalence du modèle algorithmique sous les environnements MATLAB et SMASH qui supporte VHDL-AMS. Le réseau de neurones a été décrit en tenant compte du critère de maintien de la valeur des poids synaptiques portés par des courants. Un dispositif à base de cellules de mémorisation à courants commutés a été décrit jusqu'à introduire un modèle structurel. L'ensemble des résultats obtenus a permis de valider le portage des modèles et des architectures de réseaux neuronaux dans l'environnement d'analyse, de mesurer l'influence du choix de l'algorithme d'apprentissage du réseau sur sa performance et de proposer un partitionnement en blocs élémentaires pour la réalisation de l'application.

Dimensionner les blocs élémentaires suscite de nouvelles interrogations. A partir de quels critères peut-on définir la topologie la mieux adaptée pour chacun des blocs ? Comment dimensionner de telles topologies ? Nous savons qu'un réseau de neurones est un système auto adaptatif dont les poids synaptiques évoluent et cherchent à compenser les écarts entre le signal observé en sortie du réseau et le signal souhaité. Il parait a priori impossible de déterminer les dynamiques des courants synaptiques et donc les spécifications propres à chaque cellule mémoire de courant du réseau. De plus et du fait de la complexité de l'application, le nombre de niveaux de description augmente entraînant avec lui une perte des liens explicites entre les objectifs de l'application et les contraintes de la réalisation. Aussi, le passage d'un niveau de description à un autre niveau nécessite des prises de décisions qui soient en lien avec les spécifications de l'application. Cette observation établie le trait d'union avec la troisième étape de notre démarche dont l'objectif a été de mesurer l'impact d'un modèle de circuit sur les performances de l'application. Cette mesure nous a permis de juger de la pertinence d'une topologie particulière nécessaire à l'obtention d'une réalisation robuste aux dérives de fabrication et répondant aux objectifs fixés par les spécifications de l'application.

La troisième étape de notre démarche se focalise sur le dimensionnement d'un bloc qui assure l'interface entre les signaux provenant de la matrice de pixels et les blocs internes du réseau. Ce bloc d'interface est réalisé autour d'un amplificateur à transconductance (OTA). Cette action vise à rendre notre démarche de dimensionnement généralisable à d'autres fonctions électroniques et contribue à rechercher l'efficacité des outils d'aide à la conception de circuits mixtes. Nous avons dressé un bilan des méthodes actuelles de dimensionnement et plusieurs observations nous ont guidés afin de proposer de nouvelles pistes :

- La première observation relève qu'une approche du dimensionnement par le calcul des performances à partir d'un jeu d'équations analytiques conduit à réduire les temps d'analyse et offre la possibilité d'explorer rapidement les performances de plusieurs architectures. Une telle approche permet de déterminer très tôt dans le processus de conception toute inadéquation des performances avec les spécifications générales du système.
- La seconde observation souligne l'importance du choix du point de départ du processus d'optimisation pour la recherche de la solution optimale. Un point de départ placé dans le voisinage de l'optimum augmente les chances de convergence de l'algorithme d'optimisation. Déterminer un point de départ nécessite de contraindre l'optimiseur à ne rechercher des solutions que dans le sous-espace des ajustements où toutes les hypothèses de fonctionnement du circuit sont satisfaites c'est-à-dire le sous-espace '*faisable*'. Extraire le sous-espace '*faisable*' c'est alors résoudre un CSP en considérant

l'espace des ajustements dans son ensemble afin de supprimer les points « inacceptables » qui n'auront pas à être évaluées lors de la phase d'optimisation.

- La troisième observation fait état de l'importance d'une conception robuste aux dérives de fabrication. Le rendement de production (*Yield*) est devenu aujourd'hui un enjeu majeur auprès des fabricants de circuits [ISCAS 2010, ITRS 2009]. La réduction d'échelle des dimensions des transistors ou l'accroissement de la complexité des circuits en autre provoque un taux d'erreur de fonctionnement des circuits après fabrication. On observe, de plus en plus tôt dans le cycle de conception, une prise en compte de l'influence des dispersions des performances dû essentiellement à un accroissement des dispersions des procédés de fabrication. Améliorer le rendement de production c'est alors prévoir au plus tôt dans le processus de conception les probables dérives de fabrication de paramètres technologiques.

L'idée générale de notre démarche est alors de faire précéder la phase de dimensionnement d'une phase exploratoire qui repose sur des modèles de performances décrits par un jeu d'équations. Cette phase exploratoire retourne un sous-espace *'faisable'* à partir duquel sera extrait un point d'ajustement muni d'une tolérance maximale vis-à-vis des risques de dispersions des paramètres technologiques.

Pour expliciter le sous-espace *'faisable'*, nous cherchons à répondre à un problème inverse c'est-à-dire à déterminer le sous-espace solution en partant des spécifications. L'originalité de notre approche, se base sur les méthodes de calculs ensemblistes qui offrent les outils mathématiques adaptés à la résolution de CSP en s'appuyant sur l'algorithme d'inversion ensembliste. Une telle approche permet de mener une recherche exploratoire globale des espaces d'ajustement et de performance comparativement bien plus rapide que les méthodes de recherche stochastiques [Neu-02].

L'utilisation de l'analyse par intervalles permet de résoudre de manière approchée mais garantie un problème d'inversion ensembliste. Le sous-espace *'faisable'* est alors explicité par des listes de pavés qui en donnent une approximation minorante ou majorante. Le niveau d'approximation est ensuite déterminé par un processus itératif de division des pavés qui conduit cependant à un accroissement significatif des temps de calcul.

Pour limiter le coût en ressources et en temps de calculs engendrés par le processus de division, nous privilégions l'utilisation des opérateurs de contraction de pavé. De tels opérateurs permettent de rejeter immédiatement les domaines inconsistants d'un pavé. Le processus de contraction est déterminant dans la réduction du temps d'analyse d'un sous-espace de grande dimension (n>10). Il permet de « resserrer » la zone d'incertitude qui contient la frontière laissant au processus de division des pavés de taille plus limitée. La grande force de ces opérateurs vient de leur capacité à traiter en des temps très courts des problèmes formalisés par des relations analytiques fortement non-linéaires. Un travail préparatoire est nécessaire et vise à fragmenter la complexité de la relation en une série de fonctions élémentaires permettant d'assurer le lien réciproque entre l'espace des ajustements et des performances.

Finalement trois opérations interviennent au cours de la phase d'exploration :

1 – une opération de contraction de pavé ;
2 – une opération de division de pavé ;
3 – une opération de classification des pavés.

C'est par un traitement itératif de ces trois opérations que nous avons construit notre méthode de caractérisation du sous-espace *faisable*. Une stratégie d'exploration a été proposée et se base sur :

- la possibilité de contracter l'espace de conception préalablement avant toute phase d'exploration ;
- de contracter un espace en tenant compte d'hypothèses de fonctionnement, et de critères de performances ;
- de diviser un sous-espace systématiquement dans toutes les dimensions du problème ou itérativement ;
- de diviser un pavé par bissection ou trisection ;
- de classer après une phase de contraction et/ou de division.

Notre méthode a été appliquée au redimensionnement d'un OTA en considérant sept variables d'ajustements pour dix variables de performances. Le sous-espace *faisable* a été retourné en 21 secondes.

A l'issue de la phase exploratoire, une procédure complémentaire analyse le sous-espace *faisable* afin d'en extraire un point de fonctionnement nominal présentant la plus grande marge de robustesse à l'encontre des variations des paramètres du processus de fabrication. Cette procédure cherche à inscrire le pavé aux plus grandes dimensions à l'intérieur du sous-espace formé par la réunion des pavés inclus dans le sous-espace *faisable*. En calculant le volume du pavé, il est possible d'évaluer l'aptitude d'une topologie à satisfaire les spécifications visées. Cette mesure du volume peut être avantageusement utilisée comme critère de sélection de topologie. Voyons comment une telle approche de la sélection s'inscrit dans une démarche globale de conception.

Notre travail dans une démarche globale de conception

La seconde partie de notre travail s'est focalisée sur la problématique de l'amélioration des méthodes et outils de conception de circuits mixtes.

Nous avons rassemblé grâce au VHDL-AMS les différents niveaux de description technique de notre application neuronale sous un environnement de simulation unique. Seule l'idée conceptualisant l'application a été développée et analysée séparément sous Matlab (figure 7-1). Nous avons alors proposé une méthodologie de transposition des descriptions issues de MATLAB qui soient interprétables sans erreur par le VHDL-AMS. Le VHDL-AMS offre une hiérarchisation du flot de conception par la description d'un circuit à différents niveaux d'abstraction et facilite le déplacement à travers la hiérarchie. Ce cadre d'analyse permet d'évaluer la faisabilité d'une solution technique par une comparaison des performances et des spécifications à tous les niveaux de développement du projet. Si pour une architecture donnée, la comparaison conclue à une convergence des performances vers les spécifications, une première proposition de dimensionnement est faite à l'étape ultime de dimensionnement classiquement basée sur un algorithme d'optimisation lui-même appelant le simulateur SPICE pour l'évaluation précise des grandeurs de performances.

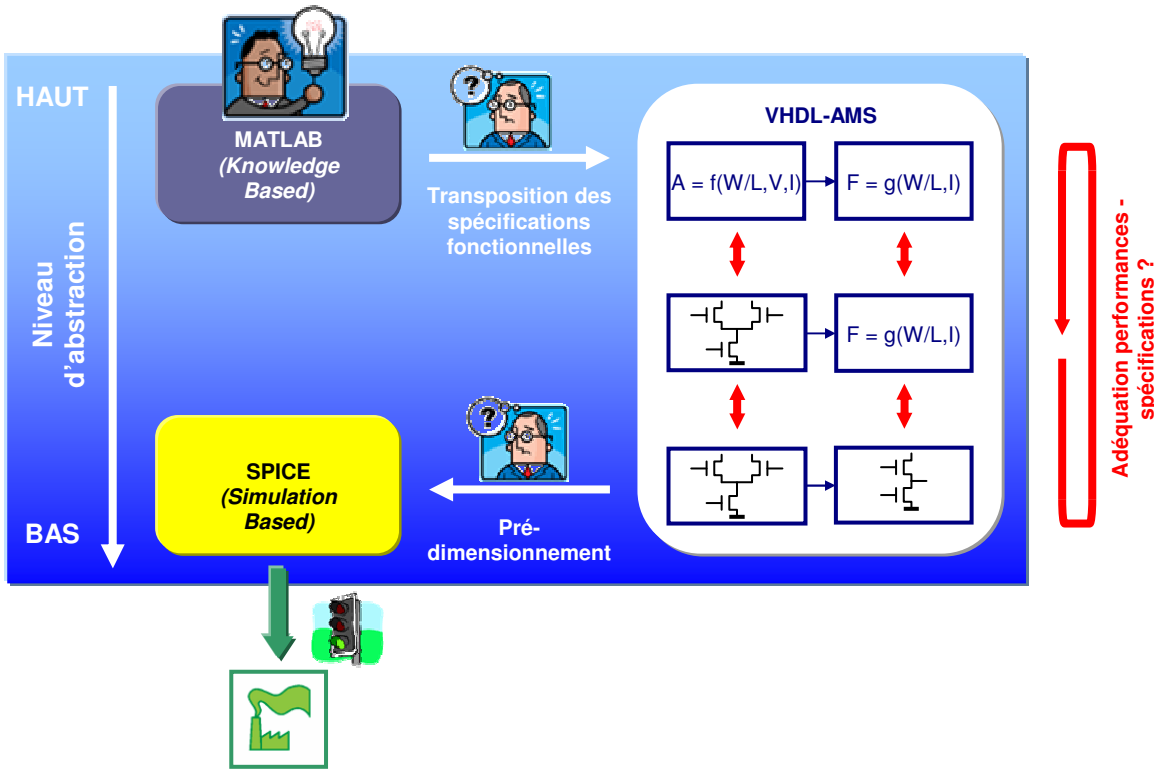


Figure 7-1 : Le processus de conception d'un circuit mixte

L'analyse d'un circuit sous l'environnement VHDL-AMS suppose qu'une architecture ait été choisie au préalable. Cette condition introduit le second axe de notre travail qui consiste à remonter des informations ou des critères utilisables par une unité de sélection de topologies. Nous avons utilisé l'outillage mathématique de Matlab appliqué à l'analyse par intervalles pour calculer ces critères de sélection. Ce constat soulève la question de la capacité du langage VHDL-AMS à proposer des fonctionnalités équivalentes à celles de Matlab. Le langage VHDL-AMS est régulièrement aménagé pour supporter de nouvelles extensions et il est probable qu'on puisse un jour disposer d'une version suffisamment complète pour mettre en œuvre notre méthode à partir de ce langage. L'idée d'ajouter un paquetage de fonctions en analyse par intervalles est réalisable mais limiterait l'analyse aux cas de spécifications statiques. Dans l'idéal, la prise en compte du calcul d'expressions différentielles à variables d'intervalles permettrait d'étendre la simulation à l'encadrement garanti de grandeurs variables dans le temps. On se retrouve alors dans le cadre de la simulation ensembliste de systèmes d'équations à temps continu sous la forme d'équations d'état :

$$\begin{cases} x' = f(x,t) \\ x \in IR^n \end{cases}$$

Aujourd'hui, de nouvelles méthodes s'appliquent au calcul d'équations différentielles ordinaires en présence d'incertitude [Ram-05] et font appel aux techniques d'intégration numérique garantie. De telles méthodes en sont encore au stade des développements méthodologiques et restent insuffisantes pour résoudre des problèmes de grande dimension décrits par des modèles à temps continu.

Une autre préoccupation concerne la formalisation et l'exécution de spécifications non fonctionnelles. Les spécifications non fonctionnelles peuvent exprimer par exemple les moyens matériels dont dispose le concepteur pour concevoir un système (contrainte d'espace mémoire par exemple) ou encore les ressources de développement (financement, qualité, temps de développement…). Or le langage VHDL-AMS est un langage de description matériel capable d'exprimer et d'exécuter des descriptions fonctionnelles ou comportementales uniquement. Une réflexion est lancée afin d'aménager le langage et d'y associer un type de spécifications [Her-a]. Il est également probable que le VHDL-AMS soit associé à d'autres formalismes situés plus en amont au profit d'une conception globale de système. Une approche système des applications impose de travailler à plusieurs, avec des compétences métiers différentes et des modèles globaux issus d'une activité pouvant être collaborative. Les normes telles que SysML [OMG] ou UML permettent l'expression mais demeurent là aussi incapables d'exécuter les spécifications d'un système. Notons que de nouveaux langages de conception mettent en relation des simulations système, matériel et logiciel. Par exemple, le langage SystemC, et bientôt SystemC-AMS [OSCI] pour les applications mixtes, étend l'homogénéisation du flot de conception de la formalisation des spécifications générales d'un système du produit à concevoir jusqu'au niveau structurel. Enfin, le langage ROSETTA [FDL'08] adopte à la fois un langage formel pour l'expression des spécifications et un langage de description fonctionnel. Sa norme est en cours de définition mais ce langage pourrait à terme se positionner comme l'outil facilitant les échanges entre les différents langages (SysML, UML, SystemC, Java, VHDL-AMS) afin d'encourager la co-simulation.

La figure 7-2 permet de situer l'impact de notre approche sur la sélection de topologie parmi le flot complet de conception d'un système. Actuellement, la formalisation graphique des spécifications est couverte par la norme SysML pour l'ingénierie système et UML dans l'application logicielle. Quant à ROSETTA, il permet de formaliser mathématiquement les spécifications. Le langage apporté par ces normes permet en autre d'aboutir un partitionnement Hardware/Software. Le software est typiquement implémenté en langage C/C++ ou JAVA et le hardware en langage de description matériel (HDL). Des passerelles existent entre le niveau de spécifications et les niveaux de description sous-jacents des parties software et hardware. UML dispose d'extensions autorisant une implémentation en C++ alors que SysML supporte également une implémentation orientée hardware.

Aujourd'hui, SysML tente de rendre exécutable les exigences par une translation bidirectionnelle SysML - VHDL-AMS. Dans le cadre de notre application, ce lien nous permettrait de relier les spécifications à la conception. Parallèlement à cela, le langage de spécification SystemC-AMS rassemble sous son formalisme les niveaux d'abstraction, software, comportemental et structurel.

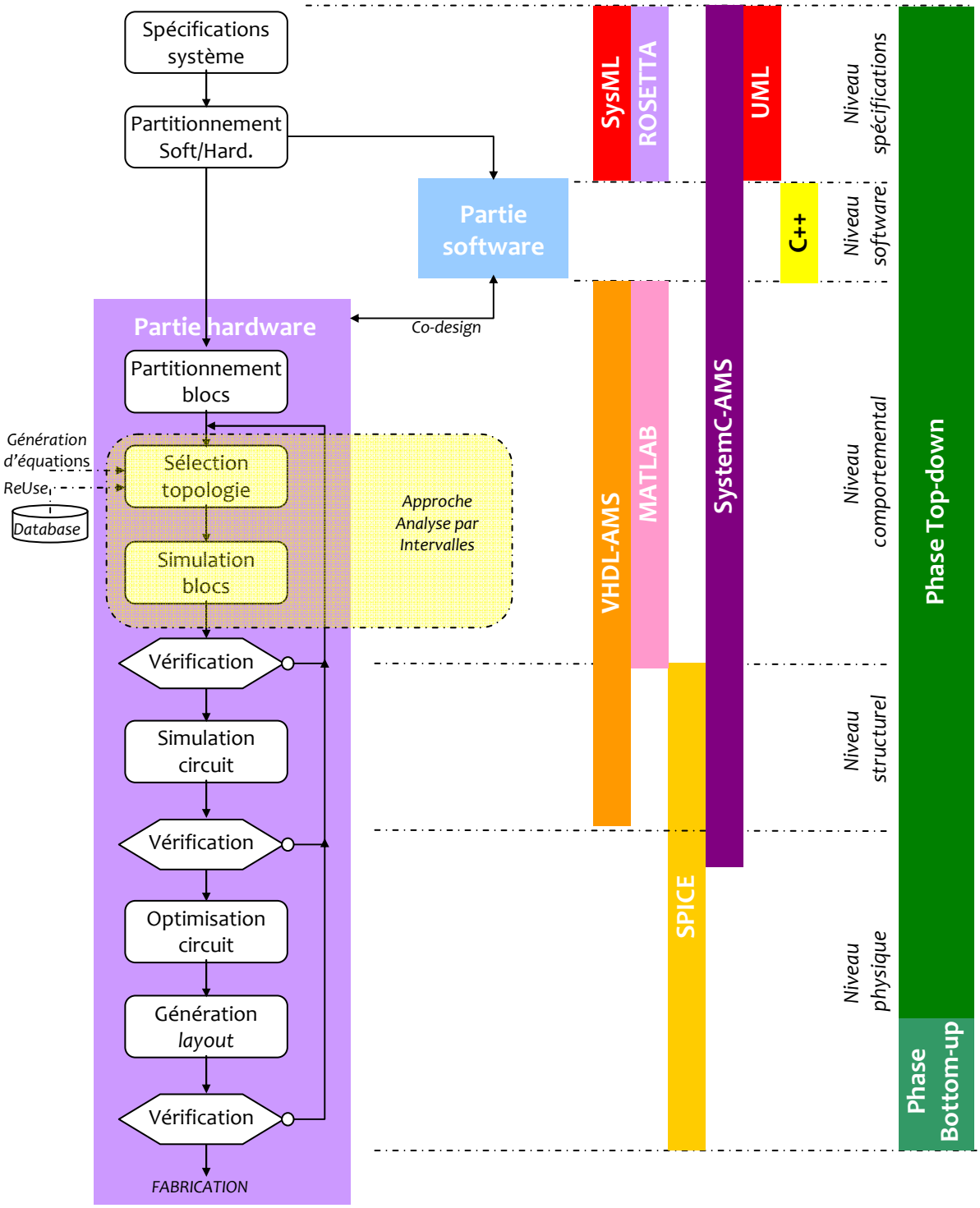


Figure 7-2 : Les outils de synthèse dans le flot de conception

L'unité de sélection de topologie repose sur une abstraction au niveau comportemental faisant le lien entre le niveau de spécifications et structurel. Ce lien est encore assuré manuellement et l'utilisation de Matlab provoque une discontinuité dans le flot de conception mettant un frein à une synthèse automatisée. L'efficacité de cette méthode est également conditionnée par la génération de modèles de topologie basée sur une approche par équations. La génération

d'équations est aujourd'hui assurée par les outils d'analyse symbolique. De tels outils nous permettent d'envisager de construire un modèle de performance analytique pour une architecture particulière en vue de proposer à terme une unité de sélection de topologie en voie d'automatisation.

A ce stade, nous pouvons identifier les points bloquants qui apparaissent à une méthodologie de conception automatisée :

La possibilité d'exprimer et de vérifier les spécifications ; ce point nous conduit à considérer la problématique de la translation de spécifications.
L'identification claire des niveaux hiérarchiques ; jusqu'à quel niveau de décomposition est-il nécessaire de décrire un système ?
La génération automatique de modèles d'équations de circuits (modèle fonctionnel, modèle de performances, …).

L'outil idéal qui se dessine possède un noyau de calcul à signaux mixtes offrant des passerelles qui permettent d'exprimer des spécifications, de les vérifier, et de pratiquer le co-design. Un tel outil doit intégrer une méthodologie qui permette d'identifier clairement les niveaux hiérarchiques. Chaque niveau doit supporter des modèles de description multi-métiers permettant :

- d'analyser et comprendre l'environnement du système,
- d'explorer, de simuler et de valider les concepts opérationnels,
- de formaliser les problème et les besoins,
- de construire les architectures fonctionnelles et physiques,
- de prévoir, de valider des comportements,
- d'estimer des performances, la fiabilité et la sûreté de fonctionnement,
- de capitaliser et de réutiliser des composants,
- de partager la connaissance, de permettre l'investigation du système.

Aujourd'hui, le langage VHDL-AMS respecte un grand nombre de ces objectifs méthodologiques en renforçant toutes les étapes de conception par l'utilisation de modèles et de simulations de prototypes virtuels. De nouvelles pistes émergent et le recours à l'évaluation directe sur des prototypes est envisagé selon un schéma couvrant le flot de conception dans son ensemble. Parmi ces pistes, le concept du « Prototypage Virtuel Fonctionnel » permet d'encadrer la conduite d'un projet à travers une démarche globale de couplage entre les différents stades d'un processus de conception.

Vers une conception automatisée basée sur le Prototypage Virtuel Fonctionnel [Her-b]

Le développement des systèmes actuels met en jeu de plus en plus de relations d'interdisciplinarité entre métiers, entre domaines physiques (électricité, optique, ou entre domaines techniques (électronique, mécanique, biologie, …). Mener à bien ce développement nécessite des moyens méthodologiques aussi bien dans la phase de définition que de vérification.
Le Prototypage Virtuel Fonctionnel rassemble grâce à la simulation, un langage de communication commun aux acteurs d'un projet pour la formalisation des spécifications globales (fonctionnelles et autres) et leurs vérifications tout au long du cycle de conception. Cette méthodologie permet de formaliser, d'échanger, de réutiliser des concepts sous la forme de modèles multi-niveaux d'un système multi-domaines.

Le cycle de Prototypage Virtuel Fonctionnel est une modification du cycle en V. Ce dernier est renforcé en privilégiant la modélisation et la simulation de chacune des étapes jusqu'à mise en œuvre d'un prototype virtuel. La figure 7-3 en donne les étapes.

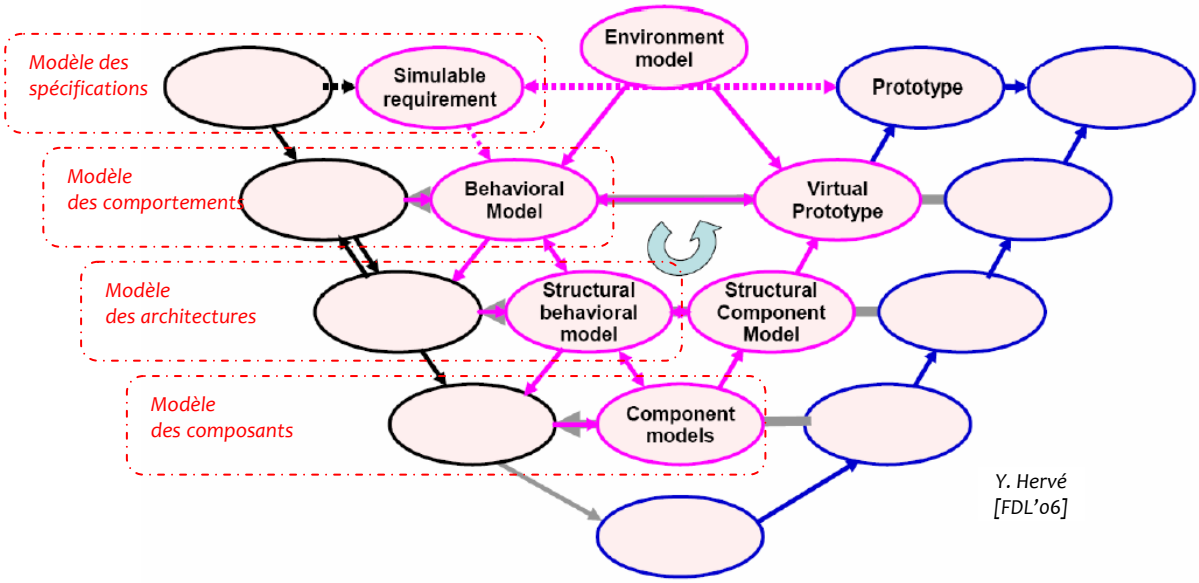


Figure 7-3 : Les étapes du cycle de Prototypage Virtuel Fonctionnel

A chacune de ces étapes, il y a comparaison des modèles comportemental et composant :
Modélisation des spécifications : Le cycle débute par la traduction des caractéristiques système de haut niveau en caractéristiques simulables. Ceci revient à translater les spécifications globales exprimées sous SysML par exemple vers un langage de description matériel comme le VHDL-AMS.
Modélisation comportementale : Il s'agit ensuite de créer un modèle comportemental du système entier. Le modèle doit retourner des performances compatibles avec les spécifications en tenant compte des contraintes environnementales (échanges thermiques, effets mécaniques, rayonnements électromagnétiques, …).
Modélisation architecturale : L'étape suivante consiste à partitionner le système en fonctions élaborées à partir de modèles comportementaux qui intègrent les contraintes de l'étape précédente. A ce stade, les modèles de hauts niveaux permettent de profiter d'une efficacité de simulation en terme de vitesse de calculs. Ces modèles pourront ensuite évoluer vers une description structurelle de plus bas niveau.
Modélisation niveau composant : les caractéristiques intrinsèques à chaque composant sont ici prises en compte. Le modèle de composant prend en considération aussi bien les caractéristiques statiques (courants de polarisation, tension de seuil, …), dynamiques (bande-passante, temps de montée, …) que technologiques (vitesse de mobilité des charges électriques, capacités parasites, …).

L'objectif du prototypage virtuel fonctionnel revient finalement à repousser le plus loin possible l'usage du prototypage physique. En d'autres termes, le but est de faire converger le cycle vers l'étape du « prototype virtuel » et par conséquent d'éviter des itérations passant par la réalisation d'un prototype physique.

Bibliographie

[Abd-94] Hervé Abdi, "Les réseaux de neurones", Collection Sciences et technologies.

[AF55] Danièle Lino, Bernard Randé, « Calcul différentiel », Techniques de l'Ingénieur, AF-55.

[Agg-07] V. Aggarwal , "Analog circuit optimisation using evolutionary algorithms and convex optimization", MIT Master report, may 2007.

[Ant-94] Antreich, K.J. Graeb, H.E. Wieser, C.U., "Circuit analysis and optimization driven by worst-case distances", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, Jan 1994, Volume: 13, Issue: 1.

[Ant-04] O. J. Antonova, D.I. Pukneva, E. D. Manolov, M. H. Hristov, "Design of CMOS OTA Core for Pratical Education", IEEE 27th Int'l Spring Seminar on Electronic Technology, 2004, p.332-337.

[Ben-99] Frédéric Benhamou, Frédéric Goualard, Laurent Granvilliers, Jean-François Puget,"Revising hull and box consistency", Proceedings of the 1999 international conference on Logic programming, Las Cruces, New Mexico, United States, pages: 230 – 244, 1999.

[Ber-05] De Bernardinis, F., Nuzzo P., Sangiovanni A., "Mixed Signal Design Space Exploration through Analog Platforms", Department of Electrical Engineering and Computer Science University of California, Berkeley, Dipartimento di Ingegneria dell'Informazione, Universita di Pisa, Italy, 2005.

[Bie-06] Michel Bierlaire, « Introduction à l'optimisation différentiable », presses polytechniques et universitaires romandes, ISBN 10 : 2-88074-669-8, édition 2006.

[Bon-05] R. Boné, H. Cardot, « Apprentissage des délais dans les réseaux de neurones récurrents. Application à la prévision de séries temporelles », 12ème Rencontres de la Société Francophone de Classification (SFC 2005), MONTRÉAL (CANADA), pp. 67-70, 30 mai - 01 juin 2005.

[Cha-92] Chang, H., Sangiovanlli-Vincentelli, A., Balarin, F., Charbon, E., Choudhury, U., Jusuf, G., Liu, E., Malavasi, E., Neff, R., Gray, P.R., "A Top-down, Constraint-driven Design Methodology For Analog Integrated Circuits", Custom Integrated Circuits Conference, 1992., Proceedings of the IEEE 1992, Issue Date : 3-6 May 1992, 8.4.1 - 8.4.6.

[Cha-01] N. chandra, G. Roberts, "Top-Down Analog Design Methodology Using Matlab and Simulink", proceedings of the IEEE International Symposium on Circuits and Systems, 6-9 May 2001, vol.5, P.139-322.

[Cle-87] Cleary, J. C., " Logical Arithmetic ", Future Computing Systems, 1987.

[Coe-96] Carlos Coello, « An Empirical Study Of Evolutionary Techniques For Multiobjective Optimization In Engineering Design », thèse, 1996.

[Dae-03] Daems W., Gielen G., Sansen W., "Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume: 22 , Issue: 5

[Dau-88] S. J. Daubert, D. Vallancourt, Y. P. Tsividis, « Current Copiers Cells», Electronics letters, 8th december1988, Vol. 24, N°25.

[Dau-90] S. J. Daubert, D. Vallancourt, « Operation and analysis of current copiers circuits», IEE Proceedings, Vol. 137, N°2, April 1990.

[Dav-87] Davis E., « Constraint Propagation with Interval Labels », Artificial Intelligence, 1987.

[Dav-94] DAVALO E. NAIM P., « Réseaux de neurones », livre éditions Eyrolles, Presse Universitaire de Grenoble, 1994.

[Deg-90] M. Degrauwe et al., "The ADAM analog design automation system," in Proc. ISCAS, 1990, pp. 820-822.

[Des-99] M. Dessouky, A. Greiner, M-M. Louërat, "CAIRO : A hierarchical layout language for analog circuits ", Proc. of Int. Conf.Mixed Design of Integrated Circuits and Systems, 1999.

[Dio-00] Analog Microelectroonic Supervised Learning Systems. FRANCESCO DIOTALEVI. Department of Biophysical and Electronic Engineering (DIBE), University of Genova. December 2000. Thesis.

[Dre-04] G. Dreyfuss. J.M. Martinez. M. Samuelides. M.B. Gordon. F. Badran. S. Thiria. L. Herault, « Réseaux de neurones – Méthodologie et applications », Deuxième édition, année 2004. Livre.

[Don-94] S. Donnay, K. Swings, G.Gielen, W. Sansen, W. Kruiskamp, D. Leenaerts, "A methodology for analog design automation in mixed-signal ASICs", European Design and Test Conference, 1994. EDAC, the European Conference on Design Automation. ETC European Test Conference. EUROASIC, the European Event in ASIC Design, Proceedings.28 Feb.-3 March 1994, p. 530 – 534.

[FDL'08] Forum for Design Languages, 2008.

[Fer-90] Fernandez, F.V. Rodriguez-Vazquez, A. Huertas, J.L. , "ASAP: a portable program for the symbolic analysis of analog integrated circuits", Dept. de Electron. y Electromagnetisme, Sevilla Univ., Jun 1990.

[Fer-97] Francisco Fernández, Angel Rodríguez-Vázquez, José L. Huertas, Georges G. E. Gielen, « Symbolic Analysis Techniques: Applications to Analog Design Automation », IEEE Press, 1997.

[Gem-92] Neural Networks And The Bias Variance Dilemna. Geman S. Bienenstock E. Doursat R. Neural Computation. Brown Unirversity Providence, USA. 1992.

[Gie-89] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: A symbolic simulator for analog integrated circuits," IEEE J. Solid-State Circuits, vol. 24, no. 6, pp. 1587-1597, Dec. 1989.

[Gie-90] G. Gielen, C. Herman, H. Walscharts, and W. Sansen, "Analog Circuit Design Optimization Based on Symbolic Simulation and Simulated Annealing", IEEE Journal Of Solid-State Circuits, Vol. 25, No. 3, June 1990.

[Gie-94] Gielen, G. Wambacq, P. Sansen, W.M., « Symbolic analysis methods and applications for analog circuits: a tutorial overview, Proceedings of the IEEE, vol. 82, issue 2, February 1994.

[Gie-02] Georges G. E. Gielen, "Modeling and Analysis Techniques for System-Level Architectural Design of Telecom Front-Ends", IEEE Transactions On Microwave Theory And Techniques, Vol. 50, No. 1, January 2002.

[Gie-05a] G.G. Gielen, "CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip", IEE Proceedings on line of Computer Digital Techniques; vol.152, n°3, May 2005, p.317-332.

[Gie-05b] Georges Gielen, Wim Dehaene, Phillip Christie, Dieter Draxelmayr, Edmond Janssens, Karen Maex, Ted Vucurevich, "Analog and digital circuit design in 65 nm CMOS: end of the road?", Design, Automation and Test in Europe, 2005, Proceedings.

[Gie-06] Georges G.E. Gielen, "Design methodologies and tools for circuit design in CMOS nanometer technologies", Solid-State Circuits Conference, 2006. ESSCIRC 2006. Proceedings of the 32nd European.

[Gra-92] H.E. Graeb, C.U. Wieser, K.J. Antreich, « Design Verification manufacturing tolerances by using worst case distances", European Design Automation Conf., DAC'92, 7-11, Sept. 1992, p;86-91.

[Han-03] E.R. Hansen, G. W. Walster, " Global optimization using interval analysis", CRC 2003 2ième édition ; ISBN : 0-82474-059-9

[Hen-07] Didier Henrion , « Polynˆomes et optimisation convexe en commande robuste », habilitation à diriger des recherches, 17 décembre 2007.

[Her-94] Hérault Jeanny, Jutten Christian, « Réseaux neuronaux et traitement du signal », livre, éditions Hermès, 1994.

[Her-a] Yannick HERVE, Ahmed FAKHFAKH, *"Requirements and Verifications through an extension* of VHDL-AMS", CNRS-PHASE Laboratory. Strasbourg, France.

[Her-b] "Functional Virtual Prototyping" Design Flow and VHDL-AMS, Dr. Yannick Hervé, INeSS-UMR 7163 ULP/CNRS laboratory, Srasbourg, Dr. Patricia Desgreys Laboratory LTCI/ENST, CNRS UMR 5141, PARIS.

[Her-02] Y. Hervé, « VHDL-AMS "Applications et enjeux industriels », Dunod 2002.

[Hui-94] Stefen Hui and Stanislaw H. *Za,* « The Widrow-Hoff Algorithm For McCulloch-Pitts Type Neurons », IEEE Transactions On Neural Networks, Vol. 5*,* No. 6, November 1994.

[Hyv-92] Hyvöven E., « Constraint reasoning based on interval arithmetic , the tolerance propagation approach", Artificial Intelligence, 1992.

[ITRS-01] Rapport ITRS 2001, "*Int. Technology Roadmap for Semiconductors", Edition 2001.*

[Jab-92] M. Jabri and B. Flower, "Weight Perturbation : An optimal architecture and learning technique for analog VLSI feedforward and recurrent multiplayer networks" IEEE trans. Neural Networks, vol.3 n°1, pp. 154-157, 1992.

[Jau-94] Luc Jaulin, "Solution globale et garantie de problèmes ensemblistes ; Application à l'estimation non linéaire et à la commande robuste", Université de Paris-sud, Thèse soutenue le 10 Février 1994.

[Jau-01a] L. Jaulin, M. Kieffer, O. Didrit et E. Walter, "Applied interval analysis, with examples in parameter and state estimation, Robust Control and Robotics", Springer-Verlag, London, 2001.

[Jau-01b] L. Jaulin, M. Kieffer, O. Didrit and E. Walter, "Applied Interval Analysis", Springer-Verlag, 2001, ISBN 1-85233-219-0.

[Jau-01c] L. Jaulin,"Le calcul ensembliste appliqué à l'automatisme", HDR, 2001.

[Joh-97] D. Johns & K. Martin, "Analog Integrated Circuit Design", John Wiley, New York, 1997.

[Kak-09] Vipan Kakkar, "Comparative Study on Analog and Digital Neural Networks ", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.7, July 2009.

[Kra-99] M.J. Krasnicki, R. Phelps, R.A. Rutenbar, L.R. Carley, "MAELSTROM : efficient simulated-based synthesis for custom analog cells", Proc. Of ACM/IEEE Design Automation Conf., 1999, pp.945-950.

[Kun-00] Kundert, K.; Chang, H.; Jefferies, D.; Lamant, G.; Malavasi, E.; Sendig, F.;, "Design of mixed-signal systems-on-a-chip", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume: 19 , Issue: 12.

[Kun-02] Daniel R. Kunkle, Chadd Merrigan, "Pulsed Neural Networks and their Application", College of Computing and Information Sciences, Rochester Institute of Technology, February, 2002.

[Lak-94] K.R. Laker, W.M.C. Sansen, "Design of analog integrated circuits and systems", McGraw-Hill internatioanl editions, ISBN 0-07-113458-1, 1994.

[Lop-06] R. Castro-López, F.V. Fernández, O. Guerra-Vinuesa, Á. Rodríguez-Vázquez "Reuse-Based Methodologies and Tools in the Design of Analog and Mixed-Signal Integrated Circuits", éditions Springer, livre 2006.

[Maa-97] Wofgang Maas, "Networks of spiking neurons: the third generation of neural network models Source", Transactions of the Society for Computer Simulation International, Volume 14 , Issue 4, December 1997.

[Mcc-07] McConaghy, T. Palmers, P. Gielen, G. Steyaert, M., "Simultaneous Multi-Topology Multi-Objective Sizing Across Thousands of Analog Circuit Topologies", Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE.

[Med-94] Medeiro, F., et al.: "A statistical optimization-based approach for automated sizing of analogue cells". Proc. ACM/IEEE Int. Conf. on Computer-Aided Design (ICCAD), 1994, pp. 594–597.

[Mic-03] Jacques MICHEL, Yannick HERVE, « Nouvelle approche de conception de réseaux neuronales analogiques », Journées Scientifiques Francophones 2003 (JSF'2003), Tozeur (Tunisia), December 20-21, 2003.

[Mic-04] J. Michel and Y. Hervé, "VHDL-AMS Behavioral Model of an Analog Neural Networks Based on a Fully Parallel Weight Perturbation Algorithm Using Incremental On- Chip Learning", Université Louis Pasteur, Strasbourg, France.

[Mil] Eric Martin, Adel Baganne, Emmanuel Casseau, « Projet MILPAT - Méthodologie et développement pour les Intellectuals Properties (IP's) pour Applications Télécom, Rapport d'avancement 1.1 », LESTER Université de Bretagne Sud.

[Moo-79] R. E. Moore, "Methods and Applications of Interval Analysis", SIAM, PA, 1979, ISBN: 0-89871-161-4.

[Mue-05] Mueller, D. Stehr, G. Graeb, H. Schlichtmann, U., "Deterministic approaches to analog performance space exploration (PSE)", Design Automation Conference, 2005. Proceedings. 42nd, Issue Date: 13-17 June 2005, On page(s): 869 - 874.

[Neu-02] A. Neumaier, "Grand challenges and scientific standards in interval analysis", Reliable Computing 8 (2002), 313-320.

[Ngu-06] Nguyen-Tuong Pierre, « Définition et implantation d'un langage de conception de composants analogiques réutilisables », thèse, 2006.

[OMG] Systems Modeling Language, "OMG SysML v. 1.2", 2010.

[OSCI] Open SystemC Initiative, "SystemC AMS extensions v1.0" , 2005.

[Phe-00] R. Phelps, M.J. Krasnicki, R.A. Rutenbar, L.R. Carley, J.R. Hellums, "Anaconda : simulation-based synthesis of analog circuits via stochastic pattern search", IEEE Trans. Computer-Aided Design, vol. 19, no. 6, pp. 703-717, June 2000.

[Por-97] J. Porte, « COMDIAC : compilateur de dispositifs actifs, reference manual », Ecole Nationale Supérieure des Télécommunications, Paris, September 1997.

[Ram-05] N.Ramdani, « Méthodes Ensemblistes pour l'Estimation », HDR, Université Paris 12, 2005.

[Rei-99] F. Reinhardt, H. Soeder, « Atlas des mathématiques », collection : Encyclopédies d'aujourd'hui, ISBN10 : 2-253-13013-3.

[Rei-03] Jérémi Regnier, « Conception de systèmes hétérogènes en Génie Électrique par optimisation évolutionnaire multicritère », thèse, 2003.

[Rut-07] R.A. Rutenbar, G.G. Gielen, J. Roychowdhury , "Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs", Proceedings of the IEEE, Volume: 95 Issue: 3 March 2007, p. 640-669, Invited paper.

[Sac-94] Sachin S. Sapatnekart, Pravin M. Vaidya, Steve M. Kang, "Convexity-based Algorithms for Design Centering", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, December 1994, Volume: 13, Issue: 12.

[Sap-93] S. Sapatnekar, P. Vaidya, S. Kang, « Convexity based algorithms for design centering », IEEE/ACM International Conference on Computer-Aided Design, ICCAD'93, Digest of Technical Papers, Nov. 93, p. 206-209.

[Ska-96] David M. Skapura, « Building Neural Networks », University of Houston at Clear Lake.

[Ste-07] G. Stehr, H. Graeb, K. Antreich, "Analog performance space exploration by normal boundary intersection and by Fourier-Motzkin elimination", I3ETCAD of Integ. Circ. & Syst.; vol.26; n°10; oct.07; p.1733-1748.

[Sun-08] Sun Q., Schwartz F., Michel J., Rom R., "Implantation study of an analog spiking neural network in an auto-adaptive pacemaker", Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA'08), Montréal (Canada), June 22-25, 2008, Proc. pp. 41-44.

[Swi-93] K. Swings, W. Sansen, "ARIADNE : Constraint-Based approach to computer-aided synthesis and modelling of analog integrated circuits", Analog Integrated Circuits and Signal Process., vol. 3, no 2, P 37-55, May 1993.

[Tan-97] N. Tan, « Switched-current design and implementation of oversampling A/D converters », livre, 1997.

[Tou-90a] C. Toumazou, J.B. Hugues, D.M. Pattullo, « Regulated Cascode Switched-current Memory Cell », Electronics letters, 8th December 1990, Vol. 26, N°5.

[Tou-90b] C. Toumazou, N.C. Battersby, C. Maglaras, « High Performance Algorithmic Switched-current Memory Cell », Electronics letters, 13th September 1990, Vol. 26, N°19.

[Tou-93] C. Toumazou, J.B. Hughes, N.C. Battersby, « Switched-currents an analogue technique for digital technology », livre, 1993.

[Val-02] Maurizio Valle, "Analog VLSI Implementation of Artificial Neural Networks with Supervised On-Chip Learning", Analog Integrated Circuits and Signal Processing, 2002.

[Vau-00] F. Vautrin, "Contibution à l'optimisation des mémoires analogiques rapides et à bas bruit dans les technologies submicroniques", Université Louis Pasteur, Strasbourg, 20 novembre 2000, Thèse.

[VSIA -99] Analog/Mixed-Signal VSI extension specification 1 version 2.0 (AMS 1 2.0), Analog/Mixed-Signal Development Working Group (VSI Alliance), 1999.

[Wal-75] Waltz, D. L., "Understanding line drawings of scenes with Shadows". In *The Psychology of Computer Vision*, pages 19–91. McGraw Hill, 1975.

[Wam-98] Piet Wambacq, Georges G. E. Gielen, and Willy Sansen , "Symbolic Network Analysis Methods for Practical Analog Integrated Circuits: A Survey", IEEE Transactions On Circuits And Systems-II: Analog And Digital Signal Processing, Vol. 45, No. 10, October 1998.

[Wil-03] Wilamowski, B.M, « Neural network architectures and learning", Industrial Technology, 2003 IEEE International Conference on Volume 1, 10-12 Dec. 2003.

[Xin-05] Xin Li Jian Wang Pileggi, L.T. Tun-Shih Chen Wanju Chiang, "Performance-centering optimization for system-level analog design exploration", Computer-Aided Design, ICCAD-2005.IEEE/ACM International Conference, Nov. 2005.

[Xu-05] Y. Xu, K-L Hsiung, X. Li, I. Nausieda, S. Boyd, L. Pileggi, "Optimization with ellipsoidal uncertainty for robust analog IC design", Design Automation Conf., DAC'2005, June 13-17, 2005.