



HAL
open science

**Modélisation et simulation d'un réseau de neurones
formels : implantation sur machine parallèle "hypercube
FPS T-40**

Djamel Benaouda

► **To cite this version:**

Djamel Benaouda. Modélisation et simulation d'un réseau de neurones formels : implantation sur machine parallèle "hypercube FPS T-40. Modélisation et simulation. Université Joseph-Fourier - Grenoble I, 1992. Français. NNT: . tel-00340978

HAL Id: tel-00340978

<https://theses.hal.science/tel-00340978>

Submitted on 24 Nov 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

présentée par

Djamel BENAOUA

pour obtenir le titre de
Docteur de l'Université Joseph Fourier - Grenoble I
(arrêté ministériel du 5 Juillet 1984)
“Mathématiques Appliquées”

**MODELISATION ET SIMULATION D'UN
RESEAU DE NEURONES FORMELS**

**Implantation sur machine parallèle
“Hypercube FPS T-40”**

Thèse soutenue le 29 Janvier 1992 devant la commission d'examen :

Président : **A. LE BRETON**

Examineurs : **M. COSNARD**
M. COTTRELL
J. DEMONGEOT
J. L. MARTIEL
J. L. SOLER

Thèse préparée au sein du laboratoire TIMB - TIM3 - IMAG.

*A mes parents,
à mes frères et sœurs
et à mes grands-parents.*

Remerciements

Je tiens à remercier tout d'abord Monsieur A. LE BRETON, Professeur à l'Université Joseph Fourier - Grenoble I, qui me fait l'honneur de présider le jury de ma thèse.

Je remercie tout spécialement mon Directeur de Recherche, Monsieur J. DEMONGEOT, Professeur à l'Université Joseph Fourier - Grenoble I, qui m'a accueilli au sein de l'Equipe Traitement d'Information et Modélisation en Biomédecine (TIMB), Institut IMAG. Qu'il soit assuré de ma profonde reconnaissance pour ses conseils, sa confiance, ses qualités humaines et scientifiques.

Je remercie Monsieur M. COSNARD, Professeur à l'Ecole Normale Supérieure de Lyon (ENSL) et Madame M. COTTRELL, Professeur à l'Université de Paris I, pour avoir accepté d'être rapporteurs de cette thèse et pour le temps qu'ils m'ont consacré.

Je voudrais remercier Monsieur J. L. SOLER, Professeur à l'Institut National Polytechnique de Grenoble (INPG), qui a bien voulu participer à ce jury. Qu'il reçoive ici mes plus vifs remerciements pour sa gentillesse et pour ses qualités humaines.

Je remercie enfin Monsieur J. L. MARTIEL, chercheur à l'INSERM, spécialiste dans la modélisation des réseaux de neurones réels, qui a bien voulu marquer par sa présence son intérêt pour la simulation des réseaux de neurones formels.

Mes remerciements vont à tous mes amis qui ont contribué de près ou de loin à la réalisation de cette thèse. Je pense particulièrement à mes amis ATIF Karim, TOUZENE Abderezak et FALLOT Pierre, pour les moments agréables que nous avons passés ensemble durant mon séjour en France. Qu'ils sachent que je leur en suis très reconnaissant.

Je remercie également ma grand-mère et mes parents pour la bonne éducation qu'ils m'ont prodiguée pendant ma jeunesse et pour leur soutien moral durant cette thèse.

Enfin, j'adresse mes remerciements à tous les membres du service de la scolarité de l'U.J.F et du service de reprographie de l'Institut IMAG.

TABLE DES MATIERES

	Page
<u>CHAPITRE I. INTRODUCTION</u>	1
I -1. Introduction générale	1
I -2. Plan général de la thèse	8
<u>CHAPITRE II. RESEAUX D'AUTOMATES ET MODELES NEURONAUX</u>	10
II -1. Introduction	10
II -2. Introduction à la neurobiologie de la cellule nerveuse	14
II -2.1 Anatomie et fonction de la cellule nerveuse	14
II -2.1.1 Les Dendrites	14
II -2.1.2 Le Soma	14
II -2.1.3 L'axone	15
II -2.2 Le rôle de la membrane basilaire	15
II -3. Réseaux d'automates et Champs aléatoires	16
II -3.1 Mesures de Gibbs	16
II -3.1.1 Définitions	16
II -3.2 Réseaux d'automates aléatoires	17
II -3.2.1 Définitions	17
II -3.2.2 Ergodicité et la mesure invariante du réseau	20
II -3.3 Champs aléatoires et processus de Harris	23
II -3.3.1 Définitions	23
II -3.3.2 Convergence	26
II -3.4 Présentation générale des réseaux à seuil	27
II -3.4.1 Définitions	27
II -3.4.2 Comportement dynamique des réseaux à seuil	29
II -4. Le modèle neuronal	34
II -4.1 Introduction	34
II -4.2 Présentation du modèle	35
II -4.3 Convergence et mesures invariantes	40
II -4.4 Estimation de la mesure de Gibbs	47
<u>CHAPITRE III. DESCRIPTION DE LA MACHINE PARALLELE : "L'HYPERCUBE [FPS] T-40"</u>	49
III -1. Introduction	49
III -2. Une vue globale sur le parallélisme	50
III -2.1 Introduction au parallélisme	50
III -3. Présentation de l'Hypercube [FPS] T-40	54
III -3.1 Généralités	55
III -3.2 Structure d'un processeur	57
III -3.3 Structure d'un nœud système	58
III -3.4 Les communications	58
III -3.5 La Mémoire	61
III -3.6 L'unité du calcul vectoriel "VPU"	62
III -3.7 Les langages	63
III -3.8 Fonctionnement de la T-serie machine	63

<u>CHAPITRE IV.</u>	IMPLANTATION DU RESEAU NEURONAU SUR LA MACHINE "HYPERCUBE [FPS] T-40" ET REALISATIONS	65
IV -1.	<i>Introduction</i>	65
IV -2.	<i>Implantation de l'algorithme</i>	66
IV -2.1	<i>Le neurone</i>	66
IV -2.2	<i>Le réseau de neurones</i>	67
IV -2.3	<i>Génération des entrées.....</i>	67
IV -2.3.1	<i>Génération d'un train de spikes</i>	67
IV -2.4	<i>Connexions et Communications</i>	75
IV -2.5	<i>Evolution des poids synaptiques</i>	84
IV -2.6	<i>Evolution du seuil</i>	86
IV -2.7	<i>Période réfractaire</i>	86
IV -2.8	<i>Fonction neurone</i>	87
IV -2.9	<i>Initialisation des paramètres de l'algorithme</i>	87
IV -2.10	<i>Récapitulation de l'algorithme du réseau</i>	88
IV -3.	<i>Réalisations</i>	89
IV -3.1	<i>Exécutions sur machine parallèle</i>	89
IV -3.1.1	<i>Comparaisons des temps d'exécution de l'algorithme</i>	90
IV -3.1.2	<i>Accélérations</i>	91
IV -3.1.3	<i>Comparaison des temps de communication</i>	93
IV -3.2	<i>Comparaisons et Résultats de Simulation</i>	94
IV -3.2.1	<i>Estimations et tests des lois asymptotiques des configurations</i>	94
IV -3.2.2	<i>Temps de convergence</i>	127
IV -3.2.3	<i>Visualisation des potentiels d'interaction</i>	137
IV -3.2.4	<i>Cliques et Amas du réseau</i>	140
IV -3.2.5	<i>Tests de validation de la mesure de Gibbs</i>	159
<u>ANNEXE 1.</u>	METHODES, TESTS ET PROGRAMMES	163
-	<i>PARTIE (A) : Méthode d'estimation et test du Khi-deux</i>	164
-	<i>PARTIE (B) : Programme en langage "C"</i>	172
<u>ANNEXE 2.</u>	VISUALISATIONS, SCHEMAS ET PROGRAMMES	185
-	<i>PARTIE (A) : Visualisation des potentiels, Cliques et Amas sur le VINIX</i>	186
-	<i>PARTIE (B) : Schémas et Programmes en langages "C" et "FORTRAN"</i>	193
<u>ANNEXE 3.</u>	PROGRAMME PRINCIPAL DE L'ALGORITHME	232
<u>BIBLIOGRAPHIE</u>	236

CHAPITRE I

INTRODUCTION

I -1. Introduction générale

Depuis quelques années, nous assistons à de grands progrès dans le domaine de l'informatique, tels que l'apparition de machines parallèles implémentées sur des circuits micro-électroniques à haute intégration comme le "VLSI", et récemment sur de nouvelles technologies, les circuits optoélectroniques, ce qui a permis d'accélérer le temps de calcul (cf. Chap. III). Ces nouvelles architectures sont très différentes des machines séquentielles ou conventionnelles, la structure fondamentale de ces dernières, conçue par J. Von Neumann dès 1948 (cf. J. Von Neumann (66)), étant fondée sur des réseaux d'automates dits cellulaires. Le but de ces recherches est de construire un jour des machines capables de résoudre des problèmes inaccessibles aux ordinateurs conventionnels ou classiques et notamment des machines spécialisées : machines capables d'imiter ou mimer certaines capacités de notre cerveau !

De nombreux travaux ont été développés dans ce sens par beaucoup de chercheurs tels que W. McCulloch et W. Pitts (43) (Chicago), F. Rosenblatt (58) (Cornell), B. Widrow et M. Hoff (60) (Stanford), puis entre les années 1960/1980, comme M. Minsky, S. Papert, S. Amari, K. Fukushima et S. Grossberg et enfin J. Hopfield (82, 84, 85) (CIT), J. A. Anderson (83) (Brown), T. Kohonen (84) (Helsinki), T. Sejnowski (84) (J. Hopkins), G. Hinton (84) (Carnegie-Mellon), C. R. Rosenberg (85) (Princeton), C. Von der Malsburg (86) (Göttingen), F. Fogelman-Soulié (85, 87) (Orsay), Y. Le Cun (87) (Paris VI), et bien d'autres, sur ce qu'ils ont convenu d'appeler "Réseaux de neurones formels" ou tout simplement sur la modélisation mathématique des systèmes biologiques de traitement de données.

Pour l'heure, les chercheurs en sont à la construction de circuits neuronaux ou de simulateurs spécialisés de réseaux de neurones de taille réduite, comportant notamment de vrais réseaux de processeurs parallèles. En particulier, des mémoires associatives ont été conçues et réalisées sur des circuits micro-électroniques ; ainsi dans les laboratoires Bell de la firme américaine AT&T (New Jersey), Joshua Alspector et ses collègues ont réalisé le premier prototype d'un réseau nommé "Machine de Boltzmann" comportant six neurones et conçu par G. Hinton, T. Sejnowski et D. H. Akley en 1984. Beaucoup d'autres ont été réalisées, notamment sur de nouvelles technologies telles que

l'optoélectronique. Bien entendu, ces derniers simulateurs sont des calculateurs analogiques, contrairement aux ordinateurs qui sont des calculateurs numériques ; cela signifie que, au lieu de traiter l'information sous forme d'impulsions électriques représentant les chiffres binaires "0" et "1", le circuit traite directement l'information contenue dans l'intensité et le sens de parcours des courants qui le traversent. En France, un réseau de neurones réalisé sous forme de circuit intégré numérique est à l'étude à l'Ecole Polytechnique et à l'ESPCI (Ecole Supérieure de Physique et de Chimie Industrielles) depuis 1987, notamment dans le cadre du projet européen lancé depuis cette date et intitulé : *BRAIN (Basic Research in Artificial Intelligence and Neurocomputing)*.

La question qu'on peut se poser est : l'approche réseaux de neurones formels est-elle mode passagère, donc destinée à disparaître comme le perceptron des années 1960, ou est-elle un bouleversement dans la manière de concevoir le traitement de l'information ? Vus les progrès enregistrés en très peu temps dans le monde, en particulier aux Etats-Unis, i.e. l'apparition des nouveaux concepts de circuits micro-électroniques et optiques, et vu le nombre croissant de chercheurs qui s'intéressent à ce domaine, on peut répondre qu'il s'agit d'une voie prometteuse.

L'étude de ces modèles que sont les réseaux de neurones formels ou artificiels, fait appel à différents domaines :

- la neurobiologie (neurone, mémoire, apprentissage, ...),
- l'informatique,
- la micro-électronique,
- les mathématiques, en particulier la théorie des probabilités.

Tous les réseaux de neurones formels s'inspirent de certaines caractéristiques des neurones réels ou biologiques : par exemple, les systèmes sensoriels tels que le système de la vision sont organisés en plusieurs couches neuronales, avec des couches de traitement intermédiaire comportant des neurones dits cachés, i.e. des neurones non reliés directement avec l'extérieur, d'où la notion de réseaux multicouches. De plus, le cerveau humain possède une autre caractéristique essentielle, autre que son organisation en couches : sa structure permet la recirculation de l'information, de telle sorte que les décisions sont prises étape par étape, i.e. le cerveau est vu comme un système dynamique déterministe ou stochastique, qui n'atteint son état d'équilibre ou attracteur qu'au bout d'un nombre fini ou non d'étapes, lorsqu'il est soumis à un stimulus extérieur. C'est cette caractéristique qui a inspiré *J. Hopfield (1982)* dans ses réseaux de neurones fortement connectés. Ces derniers sont appelés systèmes "*coopératifs*", dont les applications essentielles sont la reconnaissance de formes, l'optimisation combinatoire,...

Tout d'abord, nous allons donner brièvement quelques explications sur le fonctionnement de notre cerveau. En effet, notre cerveau est constitué d'un très grand nombre de neurones ou unités biologiques, de l'ordre de cent milliards, qui, de plus, sont extrêmement interconnectés : chacun reçoit et envoie des informations ou des signaux à plusieurs milliers de ses congénères, à l'aide de ramifications appelées "dendrites". L'information passe d'un neurone à un autre par l'intermédiaire de "synapses". Les connexions peuvent être de nature excitatrice ou inhibitrice : dans le premier cas, le neurone émetteur aura tendance à activer le neurone récepteur, et, dans le second cas, le neurone émetteur aura tendance à inhiber l'activité du neurone récepteur. Ensuite, chaque neurone prend une décision simple : à partir des influx excitateurs et inhibiteurs (somme pondérée par l'efficacité synaptique correspondante) qui lui parviennent des autres neurones, il décide de transmettre lui-même un influx soit excitateur, soit inhibiteur, et ceci en quelques millisecondes. Cette rapidité de réaction est essentiellement due au fonctionnement collectif et simultané des neurones.

Le neurone formel est un modèle particulier du neurone réel, conçu par *W. McCulloch et W. Pitts en 1943* : c'est un élément binaire dont l'état est "+1" (actif) ou "-1" (inactif). Il recalcule son état à chaque instant, en fonction des influences de tous les autres neurones formels auxquels il est relié. Autrement dit, il calcule l'influence globale, en multipliant les signaux "+1" ou "-1" issus des autres neurones, par l'efficacité synaptique correspondante, et en additionnant le tout. Enfin, il décide d'émettre un signal "+1" si la somme est supérieure à un certain seuil, et, sinon, un signal "-1" (cf. Figure 1).

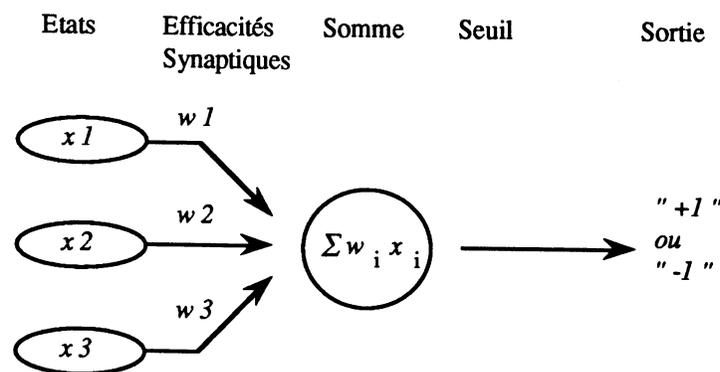


Figure 1 : Un neurone formel ou artificiel est caractérisé par son état actif "+1" ou inactif "-1", par un seuil interne et par l'efficacité de ses connexions avec ses semblables au niveau de ses synapses (analogues à celles du système nerveux). Le neurone formel recalcule son état à chaque instant, en fonction des influences de tous les congénères auxquels il est relié. Il multiplie la valeur de l'état de chacun de ses congénères par l'efficacité synaptique correspondante, et additionne le tout. Enfin, il compare cette somme à un seuil qui lui est propre et en déduit son nouvel état : "+1" si la somme est supérieure à un certain seuil, "-1" sinon.

Ainsi, chaque neurone formel est caractérisé, d'une part, par l'ensemble des efficacités synaptiques de ses liaisons avec tous les autres neurones et, d'autre part, par le seuil qui régit sa prise de décision.

La simulation sur ordinateur classique ou même sur micro-ordinateur d'un tel réseau de neurones est très simple. Il suffit, pour chaque neurone, de stocker un certain nombre de valeurs numériques : son état à un instant donné, les efficacités synaptiques qui le concernent et la valeur de son seuil interne. Quant au calcul de l'influence globale du réseau sur chaque neurone et la comparaison au seuil, ils se font d'une manière élémentaire. Le seul problème est alors d'effectuer tous ces calculs extrêmement rapidement, ce qui est pratiquement impossible sur un ordinateur conventionnel tel que : *PC, VAX, SUN*, etc..., dès que le nombre de neurones dans le réseau simulé dépasse quelques centaines. Par conséquent, un gain de temps de calcul est actuellement nécessaire. Face à cet obstacle, un bon compromis est fourni par l'utilisation des machines dites massivement parallèles telles que : "*la connection machine*", commercialisée par la société américaine "*Thinking Machine Inc.*" à 65536 processeurs, *l'hypercube [FPS]* à 32 processeurs, le *T-Node (Telmat)* à 128 processeurs, etc... Ceci nous a conduit à paralléliser notre algorithme et à l'implémenter sur la machine parallèle "*hypercube [FPS] T-40*" de *TIM3-IMAG*, comportant 32 processeurs pouvant travailler en parallèle (cf. *Chap IV*).

Ces ordinateurs parallèles se sont révélés extrêmement efficaces pour effectuer des calculs à grande vitesse. Mais ils sont beaucoup moins adaptés que le cerveau humain pour résoudre certains problèmes, par exemple reconnaître des symboles manuscrits, ce qui est une tâche facile pour un jeune enfant : seuls les ordinateurs les plus puissants réussissent à obtenir des performances acceptables. Pourquoi ? Parce que les informations ne sont pas traitées de la même manière.

Le problème crucial consiste à trouver des règles d'apprentissage permettant aux réseaux de neurones formels de remplir correctement une tâche bien définie, autrement dit de mettre au point une procédure permettant de calculer les efficacités synaptiques afin que le réseau remplisse une tâche qui lui est assignée ! l'exemple le plus simple est "*la règle de Hebb*", conçue par *D. O. Hebb* en 1949 (cf. *D. O. Hebb (49)*).

Les applications de ce que pourrait faire concrètement un tel réseau sont nombreuses : le réseau le plus spectaculaire "*Net Talk*", conçu en 1985 par *T. Sejnowski* et *C. R. Rosenberg*, qui parvient progressivement à prononcer des mots en

anglais, sans qu'on lui enseigne explicitement les règles de la phonétique ; le réseau de *Kohonen* (84), utilisé grâce à la fonction de mémoire associative, pour la classification d'images ; le réseau de système expert d'aide au diagnostic médical, mis au point par *Y. Le Cun* en 1987 ; le réseau sur la mémoire de visages, réalisé par *Rousset et al.* (88), etc... Considérons maintenant les réseaux les plus simples, par exemple ceux qui miment la capacité de ce qu'on appelle la "mémoire associative", tel celui conçu par *J. Hopfield* en 1982. Leur tâche est de mémoriser un certain nombre de "patrons", par exemple des lettres ou des chiffres manuscrits, de telle sorte que, si l'on présente au réseau une configuration détériorée, par exemple une lettre incomplète ou quelque peu déformée, il peut la restituer en faisant fonctionner collectivement ses neurones. Ce type de réseaux est notamment utilisé pour la correction automatique d'erreurs, pour la reconnaissance des formes et de la parole et les problèmes d'optimisation combinatoire.

Le réseau de neurones formels le plus simple est composé de deux couches de neurones : la première comporte des neurones d'entrée ou récepteurs, et la seconde comporte des neurones de sortie (cf. *Figure 2*). Les neurones de la première couche n'effectuent aucun traitement, leur rôle étant simplement de recevoir et de transmettre l'information à mémoriser, et ceux de la seconde couche sont capables de calculer leur état en fonction des informations provenant des neurones d'entrée et ne sont pas connectés entre eux.

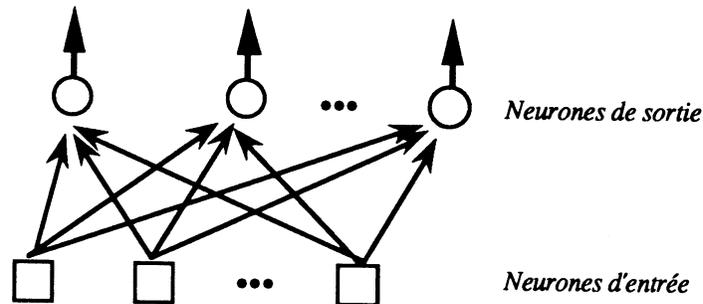


Figure 2 : Réseau de neurones simple. Les neurones (en carrés) représentent des entrées qui reçoivent les informations extérieures à identifier et d'autres neurones (en cercles) effectuent les calculs et fournissent la réponse du réseau. Ces derniers ne sont pas connectés entre eux.

Deux règles d'apprentissage ont été proposées, entre 1950 et 1960, sur ces réseaux simples : l'une d'elles, due à *B. Widrow et M. Hoff* est appelée règle de "Widrow-Hoff" ou encore algorithme de gradient, et l'autre, due à *F. Rosenblatt* est appelée règle du "perceptron". Le réseau est ainsi soumis à un apprentissage dit supervisé. Il existe aussi des réseaux dits non-supervisés, consistant à laisser le réseau évoluer en modifiant lui-même ses efficacités synaptiques selon une règle donnée, par exemple celle de *Hebb*, sans imposer aucune contrainte extérieure au réseau.

Les réseaux dits multicouches sont destinés à résoudre des problèmes plus complexes, cela par analogie à des systèmes sensoriels tels que le système de la vision, organisés en plus de deux couches : une couche de neurones récepteurs, dans la rétine par exemple, une couche de neurones moteurs reliés aux muscles et, entre les deux, des couches de traitements intermédiaires comportant des unités dites "cachées" (cf. *Figure 3*). Il s'avère impossible d'utiliser les deux règles d'apprentissage citées plus haut pour calculer les efficacités synaptiques, à cause de l'existence des unités cachées. Dès 1984, une généralisation de la règle de "Widrow-Hoff", connue sous le nom de méthode de "rétro-propagation du gradient", a permis de surmonter cet obstacle. Peu avant celle-ci, on connaissait une règle d'apprentissage du même type utilisée dans la machine de Boltzmann conçue par *Hinton, Sejnowski et Ackley*, en 1984. Cette règle consiste à imposer une configuration aux neurones d'entrées, à observer la réponse du réseau, puis à calculer les efficacités synaptiques de façon à minimiser l'écart qui sépare la réponse réelle de la réponse souhaitée, par une méthode de gradient. Ce calcul se fait couche par couche, de la sortie vers l'entrée, d'où le nom rétro-propagation.

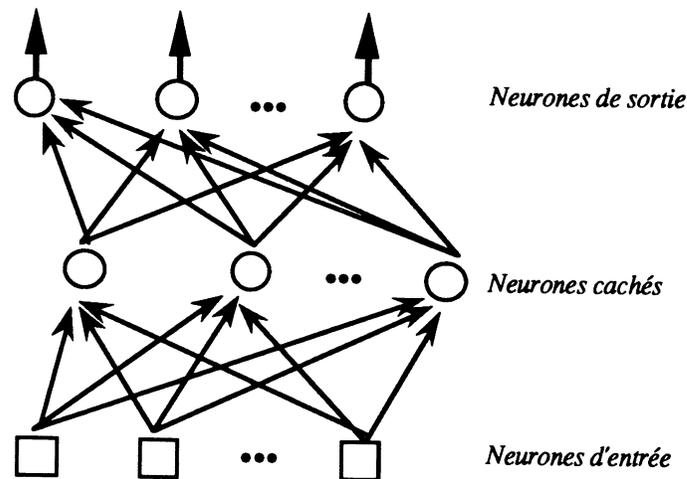


Figure 3 : Réseau multicouches de neurones formels, dans lesquels les neurones effectuant les calculs ne sont pas tous directement reliés à l'extérieur ; les neurones représentés au milieu sont des neurones cachés. Il n'existe aucune connexion entre les neurones d'une même couche.

L'un des exemples d'application les plus connus de cette méthode d'apprentissage est le réseau "Net Talk". Il y a aussi l'exemple de la "machine de Boltzmann", qui fut le premier réseau d'apprentissage à faire intervenir des neurones aléatoires ou probabilistes, i.e. à perturber le fonctionnement des neurones, à chaque étape du calcul, par l'intermédiaire d'un bruit. Dans ce réseau, les neurones ne sont pas organisés en couches ; par contre, ils sont totalement interconnectés.

L'algorithme d'apprentissage par rétro-propagation connaît à présent des améliorations et des variantes. Cependant, certains problèmes posés par les réseaux multicouches ne sont pas encore résolus, comme par exemple le choix du nombre optimal de couches cachées et d'unités par couche !

Outre son organisation en couches, le cerveau fonctionne apparemment comme un système dynamique, qui n'atteint pas instantanément un état d'équilibre lorsqu'il est soumis à un stimulus extérieur. Cette propriété a inspiré *J. Hopfield*, dès 1982. Il a proposé un modèle de réseau de neurones formels totalement interconnectés, où chaque neurone reçoit des informations de tous les autres neurones et envoie lui-même des signaux à tous les autres neurones (*cf. Figure 4*).

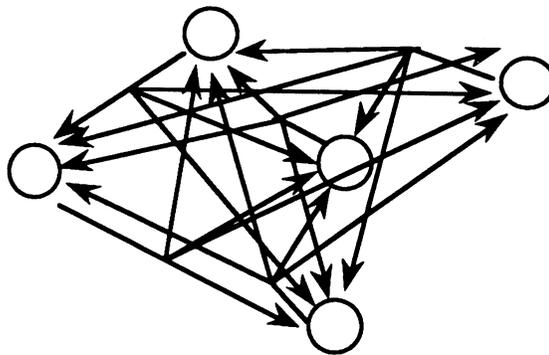


Figure 4 : Réseau de neurones d'Hopfield dit totalement interconnecté. Les neurones jouent un rôle d'entrée et de sortie.

La caractéristique essentielle du comportement du réseau de *Hopfield* réside dans l'existence d'états stables dits "*attracteurs*". En fait, lorsque la phase d'apprentissage (par exemple du type Widrow-Hoff) est achevée, i.e. lorsque le réseau a mémorisé les prototypes imposés, il est capable de reconnaître une configuration inconnue qui lui est présentée. En effet, à partir de cette configuration, les neurones recalculent leur état et le réseau évolue ainsi, en passant par des états transitoires, vers une configuration stable. Si la configuration inconnue ressemble à l'un des prototypes déjà mémorisés, le réseau aboutira à ce prototype, i.e. le prototype en question "*attire*" à lui toutes les versions incomplètes ou déformées, d'où son nom "*d'attracteur*". Il se peut que le réseau aboutisse à un état qui n'est pas l'un des prototypes, mais ressemble à l'un d'eux.

Comment cela marche-t-il ? L'idée vient de la physique, particulièrement des systèmes magnétiques désordonnés. En s'inspirant de la théorie des verres de spins, qui réside essentiellement dans l'existence d'une fonction énergie, dont les états d'équilibre correspondent aux configurations pour lesquelles cette énergie prend des valeurs minimales, *J. Hopfield* a montré qu'il est possible de décrire mathématiquement les

réseaux de neurones formels totalement interconnectés comme s'il s'agissait de verres de spins. Comme dans l'exemple des mémoires associatives, le réseau calcule ses efficacités synaptiques de telle sorte que les prototypes qu'il doit mémoriser soient des états stables attracteurs d'énergie minimum.

Cette notion d'énergie minimum a également conduit à utiliser ces réseaux, i.e. les réseaux de neurones complètement interconnectés, pour résoudre d'autres problèmes, et plus particulièrement les problèmes dits "*d'optimisation combinatoire*", qui sont caractérisés par un très grand nombre de solutions parmi lesquelles il faut trouver la meilleure, ou tout au moins l'une des meilleures. L'exemple le plus connu posé sous cette forme, par *J. Hopfield et D. Tank* en 1985, est le problème dit "*du voyageur de commerce*".

I -2. Plan général de la thèse

Cette thèse s'inscrit dans la continuité du travail de T. Hervé (*cf. T. Hervé (87)*), effectué en collaboration avec l'Institut de la Communication Parlée (ICP) de l'Institut National Polytechnique de Grenoble (INPG). Il consiste à modéliser un réseau neuronal situé en aval de la cochlée, qui constitue les premières couches de traitement des signaux de la parole issus du système auditif périphérique, plus particulièrement de la membrane basilaire. Ces signaux représentent l'information neuronale véhiculée sur des fibres du nerf auditif. L'activité de ces fibres est décrite par l'enregistrement de trains de spikes : cette activité constitue l'entrée à traiter par ce réseau. Cette entrée est modélisée par un champ aléatoire binaire bidimensionnel tenant compte des aspects temporel, spatial et stochastique du signal neuronal. Autrement dit, l'information neuronale délivrée par la membrane basilaire est non seulement temporelle, mais aussi répartie spatialement sur plusieurs fibres et, enfin, la nature stochastique du signal provient du fait que la fibre ne répond pas forcément à un même stimulus acoustique par un même train de spikes.

Le cadre général du travail présenté concerne la modélisation mathématique du réseau de neurones en question, la description de la machine parallèle à mémoire distribuées "*Hypercube FPS T-40*" utilisée comme outil de nos simulations, l'implantation du modèle neuronal sur cette machine parallèle et enfin les réalisations et interprétations de résultats de simulation. Ces travaux sont présentés en quatre chapitres comme suit :

Le premier chapitre s'inscrit dans le cadre général des réseaux de neurones, en commençant par les premiers modèles fondés sur des réseaux dits d'automates à seuil conçus par *W. S. McCulloch* et *W. Pitts* dès 1943, des réseaux dits d'automates cellulaires conçus par *J. Von Neumann* dès 1948, etc...

Le deuxième chapitre introduit la mesure de Gibbs, champs aléatoires et modèles de réseaux (déterministes et stochastiques). Puis, il présente l'étude du problème d'ergodicité des réseaux de neurones probabilistes. Dans les cas simples, notre modèle de réseau de neurones en mode d'itération parallèle (resp. séquentielle) est ergodique et converge vers une unique mesure invariante μ_P (resp. μ_S), en réponse à une entrée stationnaire. Ces mesures s'écrivent comme suit :

$$\bullet \mu_P(x) = \frac{\sum_{y \in \Gamma} e^{\langle Wx, y \rangle}}{Z} \quad \bullet \mu_S(x) = \frac{e^{\langle Wx, x \rangle}}{Z}$$

pour chaque configuration $x \in \Gamma = \{0,1\}^\Omega$, avec $\Omega = \{1, \dots, n\}$.

où $\langle \cdot, \cdot \rangle$ désigne le produit scalaire standard, $W = (W_{ij})$ la matrice $(n \times n)$ (constante dans le temps) des poids des connexions, $\Gamma = \{0,1\}^\Omega$ l'ensemble de toutes les configurations possibles de cardinal 2^n , Z la constante de normalisation et n le nombre de neurones formels.

Nous remarquons que les mesures μ_P and μ_S sont différentes pour certaines valeurs de $W = (W_{ij})$, par exemple on a :

$$\lim_{W_{11} \rightarrow -\infty} \delta(\mu_P, \mu_S) = +\infty \quad \text{où } \delta \text{ est la distance de Kullback.}$$

Le troisième chapitre concerne l'environnement technique où nos simulations de réseaux de neurones ont été effectuées. Il consiste en une description générale du principe du parallélisme et en une présentation détaillée de la machine parallèle à mémoire distribuées "Hypercube FPS T-40" (32 processeurs) du laboratoire TIM3-IMAG.

Enfin, le quatrième chapitre comprend l'implantation de l'algorithme du réseau de neurones sur la machine parallèle "Hypercube FPS T-40", l'expérimentation numérique et l'interprétation des résultats numériques. Ensuite, on a représenté graphiquement ces résultats, à l'aide de mesures statistiques adéquates résumant le comportement dynamique du réseau, sur station de travail VINIX (ordinateur spécialisé dans le traitement d'images) du laboratoire TIMB-TIM3-IMAG.

CHAPITRE II

RESEAUX D'AUTOMATES ET MODELES NEURONAUX

II -1. Introduction

C'est en 1948 que *J. Von Neumann* (cf. *J. Von Neumann (66)*) a introduit les réseaux d'automates, intitulés : "*Réseaux d'automates cellulaires*", conçus pour modéliser le problème de l'auto-reproduction : leur structure est fondée sur le déroulement séquentiel des tâches, c'est-à-dire que seuls peu d'éléments de la machine sont actifs, les autres étant en attente (cf. *chap III*). Par suite, dans les années 80, il est apparu de nouvelles machines à architecture parallèle implémentée sur des circuits à haute intégration, tels que les nouvelles technologies "*VLSI*", ce qui a permis d'accélérer le temps de calcul. Citons quelques exemples de machines à architecture parallèle : "*la connection machine*" construite dans le laboratoire d'intelligence artificielle de l'université MIT (cf. *Hillis (82)(84)*), "*la machine de Boltzman*" de l'université Carnegie Mellon (cf. *Hinton & al. (81)(84)*) ou la machine à architecture systolique de l'université Carnegie Mellon également (cf. *Kung (82) et Leiserson (83)*), etc...

Ensuite, dans d'autres domaines tels que la biologie, la physique, etc..., les chercheurs se sont trouvés confrontés à des systèmes de plus en plus compliqués, pour lesquels les méthodes heuristiques échouaient, d'où l'utilisation de méthodes type réseaux d'automates. Au point de vue mathématique, ces réseaux d'automates sont des systèmes dynamiques discrets : en espace et en temps d'évolution. Ils fournissent des solutions alternatives intéressantes par rapport aux modèles classiques fondés sur des systèmes dynamiques continus.

On représente, d'une manière formelle, les réseaux d'automates cellulaires par une application $F : E^{\Omega} \rightarrow E^{\Omega}$, où E et Ω sont respectivement l'espace d'état et l'ensemble cellulaire supposés finis, dont toutes les composantes sont identiques et définies localement par :

$$\exists V \subset \Omega, \exists f : E^V \rightarrow E$$

$$\forall i \in \Omega, \forall x \in E^{\Omega}, F_i(x) = f(x|_{i+V}), \text{ avec } F = (F_i)_{i \in \Omega},$$

où $x|_{i+V}$ est la configuration ayant même composantes que x sur $i+V$ et valant 0 en dehors de $i+V$.

Plusieurs travaux ont été consacrés à l'étude du comportement de ces réseaux (cf. *Demongeot & al. (83)*, *Wolfram (83)*, etc...), à l'existence d'un modèle de réseau

d'automates cellulaires à 2 dimensions capable de réaliser une machine de Turing universelle, connu sous le nom "*le modèle du jeu de la vie*" (cf. Berlekamp & al. (82), Conway (70)), introduit par Conway en 1970, et les réseaux booléens introduit par les biologistes Kauffman et Thomas (cf. Kauffman (72), Thomas (79)), ainsi que les réseaux de verres de spins (cf. Hopfield (82)), etc...

D'une manière générale, un réseau d'automates cellulaires est un ensemble d'automates ou sites localement interconnectés qui évoluent dans le temps (échelle discrète) en différents modes d'itérations (cf. § II -3. 4). Concernant leur dynamique d'évolution, elle est vue comme étant une succession d'états, dans le temps, de chaînes (réseau à une dimension) ou de boîtes (réseau à deux dimensions). Le système peut atteindre des cycles limites (ou comportements périodiques) ou points fixes (configurations stables). Par exemple, ces types de comportement ont été observés par la chaîne (réseau d'automates cellulaires à une dimension) du physicien S. Wolfram (cf. Wolfram (83)) qui cherchait à modéliser les propriétés d'auto-organisation de nombreux systèmes physiques, chimiques ou biologiques, ainsi que dans l'exemple du mathématicien Conway (cf. Berlekamp & al. (82), Conway (70)) qui a proposé le modèle de "*jeu de la vie*" fondé sur des règles simples (réseau d'automates cellulaires à deux dimensions), cherchant à modéliser la dynamique des populations d'organismes vivants.

Voyons ensuite la classe des réseaux d'automates à seuils. Les premiers pionniers du domaine, McCulloch & Pitts (cf. McCulloch & Pitts (43)), avaient introduit, en 1943, les premières notions "*d'automates à seuils*" dans le cadre de la modélisation de l'activité de la cellule nerveuse (ou neurone). En effet, ils modélisaient le neurone (ou la cellule nerveuse) par un neurone artificiel (ou formel), i.e. un automate à seuil dont les entrées représentent les signaux issus par d'autres neurones, et le seuil représente le seuil d'excitabilité du neurone en question. L'écriture mathématique de l'équation d'évolution est la suivante : un automate à seuil est une application $F : \{0,1\}^n \rightarrow \{0,1\}^n$ avec

$$\forall x \in \{0,1\}^n, \quad F(x_1, \dots, x_n) = \begin{cases} 1, & \text{si } \sum_{j=1}^n a_j x_j \geq \theta \\ 0, & \text{si } \sum_{j=1}^n a_j x_j < \theta \end{cases}$$

où $\theta \in \mathbb{R}$ est le seuil d'excitabilité du neurone, $a_j \in \mathbb{R}$ sont les poids des synapses qui relie le neurone avec les autres neurones "*j*" (si $a_j \leq 0$, la liaison est inhibitrice ; sinon, la liaison est excitatrice) et n est le nombre total des entrées issues des autres neurones.

Le premier réseau de neurones formels ou automates à seuil était destiné à résoudre des problèmes simples, par exemple à mimer la fonction de mémoire associative dont fait preuve notre cerveau. Cette dernière consiste à faire mémoriser un certain nombre d'informations, par exemple des lettres, chiffres manuscrits, etc..., par un tel réseau, et ensuite le réseau est capable de retrouver ces informations ou identifier des informations incomplètes. Ce réseau comporte deux couches de neurones formels. La première couche contient des neurones dits d'entrée ou récepteurs, leur rôle consistant à recevoir et à transmettre l'information à mémoriser. Les neurones de la deuxième couche sont capables de calculer leur propre état en fonction des informations provenant des neurones récepteurs, mais ne sont pas connectés entre eux : ce sont eux qui contribuent à la fonction de décision du réseau.

La question qui se pose est : comment le réseau réalise-t-il la fonction de mémoire associative ? ou comment s'assurer qu'il remplisse correctement cette tâche ? D'une manière générale, le problème crucial consiste à trouver des règles d'apprentissage adaptées permettant aux réseaux de neurones formels de remplir une tâche bien définie. Précisément, il consiste à mettre au point une procédure permettant de calculer les efficacités synaptiques. Dès 1949, le neurophysiologiste *D. O. Hebb* (cf. *D. O. Hebb* (49)) a émis l'hypothèse que les neurones réels ou biologiques s'auto-organisent dans leurs connexions sous l'effet des stimuli qu'ils reçoivent. Cette auto-organisation repose sur un principe simple : l'efficacité d'une synapse activatrice (resp. inhibitrice) se renforce lorsque les neurones qu'elle relie ont tendance à être actifs en même temps, c'est-à-dire en phase (resp. en opposition de phase) ; elle s'atténue dans le cas contraire.

Ensuite, dans les années 1950/1960, trois chercheurs ont introduit deux règles d'apprentissage adaptées à de tels réseaux. L'une d'elle, conçue par *B. Widrow et M. Hoff*, est appelée règle de " *Widrow-Hoff* ". Et l'autre, conçue par *F. Rosenblatt*, est appelée règle du " *perceptron* ". Le réseau est ici soumis à un apprentissage dit supervisé. En quoi consiste-t-il ? Dans la première règle, il s'agit de modifier les efficacités synaptiques de tous les neurones par un algorithme type gradient ; dans la deuxième règle, il s'agit de modifier les efficacités synaptiques des neurones qui fournissent une réponse incorrecte par un mécanisme type punition.

Ces réseaux étaient censés résoudre des problèmes simples. Pour les problèmes plus complexes, une solution consiste à organiser la décision en plusieurs étapes, ce qui revient un réseau à plusieurs couches. Le perceptron est le premier réseau organisé en trois couches : deux couches entrée/sortie et une couche de traitement intermédiaire contenant des neurones dits cachés, conçu par *Minsky & Papert* (cf. *M. Minsky & S. Papert* (69)) pour la reconnaissance de formes. Dans un premier temps, on ne pouvait

pas utiliser les deux règles citées plus haut, à cause de l'existence des couches de traitement intermédiaires.

Après l'échec du perceptron (*cf. Minsky & Papert (69)*), dans les années 1970, l'Intelligence Artificielle s'est développée, qui a pour but justement de résoudre les problèmes imposés par les performances des ordinateurs disponibles. De nombreux chercheurs ont investi dans ce domaine pour se dégager de certains obstacles imposés. Ils ont proposé des architectures très proches de la philosophie du perceptron. Certains d'entre eux, en s'inspirant de la physique du solide, utilisent les concepts de la théorie des verres de spins et de la mécanique statistique (*cf. R. Maynard et al. (81)*), et proposent des architectures capables de réaliser des fonctions de mémoire associative (*cf. J. Hopfield (82), G.E.Hinton et al.(81), F. Fogelman et al. (85, 87)*), de satisfaction de contraintes (*cf. G.E. Hinton et al. (84)*), des problèmes d'optimisation combinatoire (*cf. J. Hopfield & D.Tank (85)*), etc...

Enfin, citons les travaux *J.P. Changeux et al. (cf. Changeux (84))* dans le domaine de la neurophysiologie, ainsi que la réalisation de mémoires associatives, de machines de reconnaissance, de classification (*cf. T. Kohonen (84), ...*).

Parallèlement, les études théoriques sur les réseaux d'automates à seuil s'approfondissent (*cf. Goles (80)(82), Goles & Olivos (81), Goles & Tchunte (83), Fogelman (85), Robert (86), etc...*). L'une des propriétés les plus utilisées dans la pratique est la suivante :

soit $F : \{0,1\}^n \rightarrow \{0,1\}^n$ un réseau à seuil, de matrice de poids synaptiques $A=(a_{ij})$, alors le réseau à dynamique d'itération séquentielle n'admet que des points fixes (limites), à condition que A soit symétrique à éléments diagonaux non-négatifs (*cf. Goles & Olivos (81)*). Ces points fixes sont utilisés dans les machines étudiées par *Hopfield (cf. Hopfield (82))*, *Huberman (cf. Huberman (84))*.

Dans ce chapitre, nous présentons une introduction neurobiologique concernant la cellule nerveuse (anatomie, fonctions,...), et quelques rappels et résultats théoriques : mesure de Gibbs, réseaux d'automates, champs aléatoires et étude du comportement dynamique des réseaux d'automates à seuil. Enfin, nous étudions un modèle de réseau de neurones situé en amont du nerf auditif, qui représente les premières couches de traitement de l'information (signaux de la parole) issue de la cochlée faisant partie du système de perception périphérique de la voie auditive. Cette information est modélisée par un champ aléatoire binaire bidimensionnel tenant compte des aspects temporel, spatial et stochastique du signal neuronal.

II -2. Introduction à la Neurobiologie de la Cellule Nerveuse

II -2.1 Anatomie et Fonction de la Cellule Nerveuse

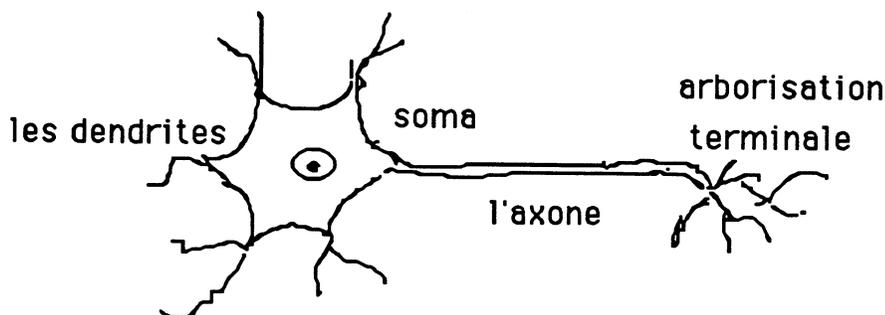


Figure 1: La cellule nerveuse (ou neurone). Le neurone reçoit des signaux provenant de plusieurs milliers de ses semblables par ses dendrites, il décide d'émettre ou non un signal à ses semblables le long de son axone. Les points de contact entre deux neurones sont appelés synapses.

Dans le cerveau humain, il y a environ 100 milliards de neurones multiformes (cf. Delmas (70), Figure 1). La cellule nerveuse est constituée de trois parties :

- Les dendrites,
- Le soma (corps cellulaire),
- L'axone.

II -2.1.1 Les Dendrites

Ce sont de fines extensions tubulaires, se terminant par des synapses. Grâce à elles, le neurone collectionne des signaux provenant des neurones voisins. En effet, ces synapses libèrent une substance chimique (neuro-médiateur) qui provoque une variation locale du potentiel membranaire. Pour l'ingénieur, tout se passe comme si le signal transmis par une synapse était pondéré par un coefficient positif ou négatif que l'on appelle "*le poids synaptique*". Si ce coefficient est positif, cela signifie que la synapse considérée a tendance à exciter le neurone (post-synaptique), sinon le neurone va être inhibé. Le signal transmis se déplace ensuite vers le soma .

II -2.1.2 Le Soma

Il renferme tous les éléments biochimiques de la cellule. Il reçoit des signaux électriques issus des autres neurones à travers les dendrites par l'intermédiaire des synapses, puis il les traite jusqu'à ce qu'une sommation dite spatio-temporelle s'opère.

Le neurone délivre alors une impulsion dite "*potentiel d'action*" ou "*spike*" une fois que cette somme franchit un certain seuil. Si c'est le cas, il sera impossible (physiologiquement) pour le neurone de décharger à nouveau pendant une certaine période qui suit l'émission d'un spike. Cette période est dite réfractaire. Enfin, ce potentiel d'action se propage le long de l'axone.

II -2.1.3 L'axone

Il est le prolongement du soma. Il se termine par des arborisations terminales qui vont établir des connexions avec les autres neurones suivants. Pour l'ingénieur, l'axone est comme une ligne de transmission sans perte. En effet, l'apparition d'un potentiel d'action venant du soma traverse l'axone à une vitesse constante et à une amplitude constante.

II -2.2 Le Rôle de la Membrane Basilaire

En réponse à un stimulus acoustique, le message sonore va franchir la voie auditive jusqu'à la membrane basilaire située à l'intérieur de la cochlée (cf. Figure 2). Ce même message va subir ensuite une véritable analyse fréquentielle par cette membrane. En effet, le long de cette dernière, il existe des récepteurs sensitifs qui convertissent les vibrations en information électrochimique déclenchant les influx nerveux dans les fibres du nerf auditif.

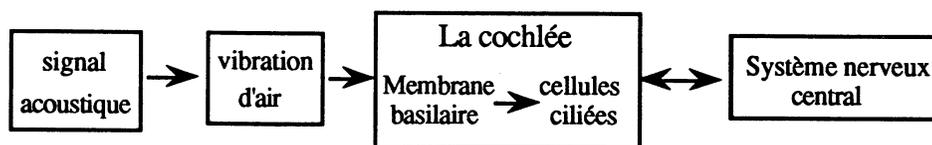
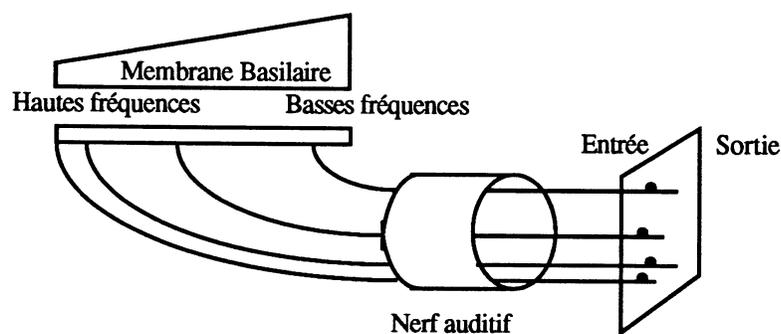


Figure 2 : Traitement de la parole (Système Auditif Humain)

II -3. Réseaux d'automates et Champs aléatoires

II -3.1 Mesures de Gibbs

II -3.1.1 Définitions

Définition 3.1. : *Mesure de Gibbs*

Une mesure de probabilité μ sur $\{0,1\}^\Omega$ est appelée mesure de Gibbs associée au potentiel U , définie de $\{0,1\}^\Omega$ vers \mathbb{R} avec $U(\emptyset) = 0$, si l'on a :

$$\mu : \{0,1\}^\Omega \rightarrow \mathbb{R}$$

$$X \mapsto \mu(X) = \frac{e^{U(X)}}{Z}, \text{ où}$$

$$Z = \sum_{Y \subset \Omega} e^{U(Y)} \text{ est la constante de normalisation.}$$

μ est encore appelée l'état de Gibbs associé au potentiel U .

Définitions 3.2. : *Potentiel de Gibbs et Potentiel d'interaction*

Un potentiel de Gibbs est une application de $\{0,1\}^\Omega$ vers \mathbb{R} ; si U est un potentiel sur $\{0,1\}^\Omega$, alors on définit le potentiel d'interaction J_U standard comme étant une application :

$$J_U : \{0,1\}^\Omega \rightarrow \mathbb{R}$$

$$X \mapsto J_U(X) = \sum_{Y \subset X} (-1)^{|X \setminus Y|} U(Y), \text{ avec}$$

$$U(Y) = \log[\mu(Y)] - \log[\mu(\emptyset)], \text{ où } \mu \text{ est la mesure de Gibbs correspondante.}$$

Inversement, par la formule d'inversion de Mœbius (cf. Barra (81)), on a :

$$U(X) = \sum_{Y \subset X} J_U(Y), \text{ avec } J_U(\emptyset) = 0.$$

II -3.2 Réseaux d'automates aléatoires

II -3.2.1 Définitions

Définition 3.3. : Automate Aléatoire

Un automate aléatoire A est un triplet d'ensembles (E, X, Y) associé à une paire de fonctions (F, G) , tels que :

$$A = [(E, X, Y), (F, G)], \text{ où}$$

- E est l'ensemble des états
- X est l'ensemble d'entrées
- Y est l'ensemble des sorties
- F est une application telle que :

$$F : X \times E^2 \rightarrow [0,1]$$

$$(x, p, q) \mapsto F(x, p, q),$$

F est la probabilité de passer de l'état " p " à l'état " q ", quand l'entrée est " x ", vérifiant :

$$\forall x \in X, \forall p \in E, \text{ on a } \sum_{q \in E} F(x, p, q) = 1$$

- G est une application telle que :

$$G : E \times Y \rightarrow [0,1]$$

$$(p, y) \mapsto G(p, y),$$

G est la probabilité d'avoir " y " comme sortie correspondant à l'état " p ", vérifiant :

$$\forall p \in E, \text{ on a } \sum_{y \in Y} G(p, y) = 1$$

Définition 3.4. : Réseaux d'automates aléatoires

Un réseau d'automates aléatoires R est un ensemble fini de sites (ou cellules) Ω associé à une paire de fonctions (S, T) , tels que :

$$R = [\Omega, (S, T)], \quad \text{où}$$

- S est une application telle que :

$$S : \Omega \rightarrow \Sigma$$

$$\omega \mapsto S(\omega),$$

est l'automate associé au site $\omega \in \Omega$, tel que :

$$S(\omega) = [(E, X_{\omega}, Y), (F_{\omega}, G_{\omega})]$$

- T est une application telle que :

$$T : \Omega \times Y^{\Omega} \rightarrow X$$

$$(\omega, f) \mapsto T(\omega, f),$$

est l'entrée enregistrée au site ω , quand chaque site $u \in \Omega$ donne une sortie $f(u) \in Y$, où $S(\omega)$ est définie dans la définition 3.3.

CAS PARTICULIER : Réseaux d'automates aléatoires binaires

On prend : $E=Y=\{0,1\}$, $\Omega \subset \mathbb{Z}^d$ (fini), donc on peut identifier l'ensemble $\{0,1\}^{\Omega}$ à $P[\Omega]$ (i.e. l'ensemble des parties de Ω) et $X_{\omega}=\{0,1\}^{N(\omega)}$, où $N(\omega)$ est l'ensemble des voisins du site ω .

On définit donc le réseau R comme suit :

1) $\forall \omega \in \Omega$, $S(\omega) = [(E, X_{\omega}, Y), (F_{\omega}, G)]$, où $\forall \omega \in \Omega$, $G_{\omega} = G$, vérifiant : pour tout $p, y \in \{0,1\}$,

$$G(p, y) = \begin{cases} 1, & \text{si } p = y \\ 0, & \text{si } p \neq y \end{cases}$$

et $F_{\omega}(D, p, q)$, la probabilité de transition de l'état " p " vers l'état " q ", quand l'entrée est $D \subset N(\omega)$, est définie par un certain potentiel U sur $\{0,1\}^{\Omega}$ comme suit :

$$\forall p \in \{0,1\}, \forall D \subset N(\omega), F_{\omega}(D, p, q) = \frac{e^{U(D \cup |q\omega|)}}{e^{U(D \cup |q\omega|)} + e^{U(D \cup |(1-q)\omega|)}},$$

$$\text{où } D \cup |q\omega| = \begin{cases} D \cup |\omega|, & \text{si } q = 1 \\ D \setminus |\omega|, & \text{si } q = 0 \end{cases}$$

2) $T(\omega, f) = \{f(u)\}_{u \in N(\omega)}$

si $N(\omega) = \{\omega_j\}_{j=1, n}$, alors $T(\omega, f) = \{f(\omega_1), \dots, f(\omega_n)\}$.

Ensuite, nous nous intéressons à l'évolution du réseau R dont la dynamique d'itération est séquentielle telle que :

le processus d'évolution associé au réseau R est un processus de Markov sur $P[\Omega]$. Soit $\{\omega_1, \dots, \omega_{|\Omega|}\}$ un ordre sur Ω et soit $f_k \in \{0,1\}^\Omega$ l'état du réseau à l'itération k (où $k \in \mathbb{N}$) (autrement dit c'est la configuration du réseau à l'étape k) ; alors, l'état du réseau R à l'itération $k+1$ est donnée, selon la règle de transition, par le système suivant :

$$f_{k+1}(\omega_i) = \begin{cases} f_k(\omega_i), & \text{si } i \neq (k+1) \bmod |\Omega| \\ \text{tiré selon "RTL"}, & \text{si } i = (k+1) \bmod |\Omega| \end{cases}$$

où "RTL" est la Règle de Transition Locale, définie par la probabilité de transition suivante :

$$P_{\omega_i} [T(\omega_i, f_k), f_k(\omega_i), \cdot].$$

Ainsi, l'ordre choisi sur la boîte Ω détermine un parcours de ω_1 à $\omega_{|\Omega|}$, chaque itération de ce parcours correspondant au changement d'un site (et un seul) de Ω associé à cette itération.

Le sous-processus associé au réseau R , correspondant aux itérations $0, |\Omega|, 2|\Omega|, \dots, k|\Omega|, \dots$, est un processus de Markov homogène qui admet, asymptotiquement, une unique mesure invariante μ , telle que (cf. Proposition 3.2., ci-après) :

$$\mu(X) = \frac{e^{U(X)}}{Z}, \text{ où } Z \text{ est la constante de normalisation.}$$

c'est-à-dire que μ est solution du système d'équation $\mu = \mu.M$, ou encore que μ est le vecteur propre à gauche de la matrice de transition M du processus de Markov associé à la valeur propre $\lambda=1$.

II -3.2.2 Ergodicité et Mesure Invariante du Réseau

Proposition 3.1. : *Le processus de Markov associé au réseau R pris au temps $0, |\Omega|, 2|\Omega|, \dots, k|\Omega|, \dots$ est ergodique.*

Preuve : D'après Demongeot (85), soit M la matrice de transition du processus $(X_{k/|\Omega|})_{k \in \mathbb{N}}$ associé au réseau R , telle que :

$$M = \prod_{i=1}^{|\Omega|} M_{\omega_i}$$

où M_{ω_i} est la matrice de transition de l'itération $i-1$ à l'itération i .

Supposons que les éléments de $P[\Omega]$ sont rangés suivant l'ordre lexicographique obtenu à partir de l'arrangement $\{\omega_1, \dots, \omega_{|\Omega|}\}$ choisi sur Ω .

$(M_{\omega_i})_{P,Q}$ est le terme général de la matrice de transition M_{ω_i} . Il représente la probabilité de passer de l'état P , à l'itération $i-1$, à l'état Q , à l'itération i , telle que :

$$(M_{\omega_i})_{P,Q} = \begin{cases} 0, & \text{si } |P \Delta Q| > 1 \\ F_{\omega_i} [P \cap N(\omega_i), \mathbb{1}_P(\omega_i), \mathbb{1}_Q(\omega_i)], & \text{sinon} \end{cases}$$

Par conséquent, le terme général de la matrice M s'écrit comme suit :

$$M_{P,Q} = \prod_{i=1}^{|\Omega|} (M_{\omega_i})_{P_i, P_{i+1}}, \quad \text{où}$$

$$P_1 = P$$

⋮

$$P_i = (P \setminus ((P \setminus Q) \cap A_i)) \cup ((Q \setminus P) \cap A_i), \text{ avec } A_i = \{\omega_1, \dots, \omega_{i-1}\}; |\Omega| > i > 1$$

⋮

$$P_{|\Omega|} = Q$$

Or, pour tout $P, Q \subset \Omega$, $M_{P,Q} > 0$; d'où, d'après le théorème de Perron-Frobenius (cf. Karlin et al. (75)), le processus $(X_{k/|\Omega|})_{k \in \mathbb{N}}$ est ergodique.

Proposition 3.2. : L'unique mesure invariante μ du processus de Markov $(X_{k/\Omega})_{k \in \mathbb{N}}$ associé au réseau R est la mesure de Gibbs associée au potentiel U .

Preuve : D'après Demongeot (85), puisque M est irréductible et primitive, à cause de sa stricte-positivité (cf. Gantmacher (59)), alors il existe une unique mesure invariante qui correspond au comportement asymptotique du réseau R . Pour cela, il suffit de vérifier que la mesure de Gibbs associée au potentiel U est invariante pour M , donc pour chaque facteur M_{ω_i} .

En effet, par définition de F_{ω_i} , on a :

$$\begin{aligned} (M_{\omega_i})_{P, P \cup \omega_i /} &= F_{\omega_i} [P \cap N(\omega_i), \mathbb{1}_{P(\omega_i)}, 1] \\ &= \mu [|\omega_i| \mid P \cap N(\omega_i)] \end{aligned}$$

où μ est la mesure de Gibbs associée au potentiel U , comme celui de F_{ω_i} .

Et aussi on a :

$$(M_{\omega_i})_{P, P \setminus \omega_i /} = 1 - (M_{\omega_i})_{P, P \cup \omega_i /}$$

d'où μ est invariante pour chaque M_{ω_i} , donc pour M .

• LA VITESSE DE CONVERGENCE DU PROCESSUS :

On définit l'entropie "H" de Kolmogorov-Sinai du processus de Markov $(X_{k/\Omega})_{k \in \mathbb{N}}$, associé au réseau R , de matrice de transition M , par :

$$H = - \sum_{P \subset \Omega} \mu(P) \sum_{Q \subset \Omega} (M)_{P, Q} \log [(M)_{P, Q}] ,$$

où μ est la mesure asymptotique du réseau R .

Etant donnée μ_0 une mesure de probabilité initiale sur $P[\Omega]$, alors μ_k est obtenue à l'itération $k/\Omega/$ du processus de Markov de matrice de transition M , par :

$$\mu_k = \mu_0 M^k ,$$

où μ_0 est un vecteur transposé $2^{|\Omega|}$ -dimensionnel. On déduit ensuite une borne de la vitesse de la convergence du processus en appliquant la proposition suivante :

Proposition 3.3. : Si δ désigne la distance de Kullback sur les mesures de probabilités sur $P[\Omega]$, alors il existe un λ ($\lambda > 0$) et K dans \mathbb{N} assez grand, tel que :

$$\delta(\mu_0 M^k, \mu) \leq \lambda k H ; \quad \forall k > K$$

Preuve : c'est une application directe de Goldstein (81) et Tuljapurkar (82).

La connaissance de l'entropie " H " donne une borne supérieure de la vitesse de convergence vers la mesure asymptotique μ (la mesure invariante). Malheureusement, il est difficile de calculer cette entropie dès que la taille du réseau est importante de l'ordre de 32×32 (cas où $\Omega \subset \mathbb{Z}^2$). Par conséquent, on ne peut exploiter le résultat donné par la proposition ci-dessus.

Remarque 3.1. : Le problème de passage à la limite thermodynamique consiste à étudier les mesures limites obtenues en faisant tendre Ω vers \mathbb{Z}^d (i.e. un réseau infini). On dit qu'il y a une transition de phase, si le cardinal des mesures limites est strictement supérieur à 1 (cf. Spitzer, Preston, Ruelle, Dobrushin, Landford).

II -3.3. Champs Aléatoires et Processus de Harris

II -3.3.1 Définitions (D'après Demongeot (85))

Définition 3.5. : Champ Aléatoire

Un champ aléatoire C sur Ω est une famille de variables aléatoires $\{C_\omega\}_{\omega \in \Omega}$, à valeurs dans E , telle qu'il existe un espace de probabilité $(\Lambda, \mathcal{A}, P)$, avec :

$$C = \{C_\omega\}_{\omega \in \Omega} : (\Lambda, \mathcal{A}, P) \rightarrow (E^\Omega, \mathcal{B}(\Omega), P_C),$$

où $\mathcal{B}(\Omega)$ est la tribu borélienne sur E^Ω engendrée par les cylindres finis (cf. Fricot (85)) et P_C est la mesure canonique du vecteur aléatoire $\{C_\omega\}_{\omega \in \Omega}$.

Remarque 3.2. : Champ aléatoire binaire

Si le champ aléatoire est binaire, on prend $E = \{0,1\}$; de plus, si Ω est un sous-ensemble fini de \mathbb{Z}^d , $\mathcal{B}(\Omega)$ est l'ensemble de tous les sous-ensemble de $\{0,1\}^\Omega$, i.e. $\mathcal{B}(\Omega) = P[\{0,1\}^\Omega]$. Ensuite, on confond l'espace canonique $\{0,1\}^\Omega$ avec $P[\Omega]$, dès que Ω est fini.

D'après le théorème de Hammersley-Clifford (cf. Besag (74)), pour chaque mesure canonique de probabilité P_C ($P_C > 0$) d'un champ aléatoire C sur Ω (fini), P_C est la mesure de Gibbs associée à un certain potentiel U , défini de $P[\Omega]$ vers \mathbb{R} .

Remarque 3.3. : (cf. Proposition 3.1. ; cf. Remarque 3.2.), on dit que le réseau R réalise asymptotiquement le champ aléatoire $\{C_\omega\}_{\omega \in \Omega}$.

Définition 3.6. : Critère Markovien spatial

Un champ aléatoire binaire est un champ Markovien spatial, si sa mesure canonique P_C vérifie :

- 1) P_C est strictement positive sur $\mathcal{B}(\Omega)$
- 2) $\forall \omega \in \Omega, \forall A \subset \Omega, P_C(\omega | A) = P_C(\omega | A \cap N(\omega))$,
où $N(\omega)$ est l'ensemble des voisins de ω .

Définition 3.7. : Critère de renouvellement spatial

Un champ aléatoire binaire est un champ de renouvellement spatial, si sa mesure canonique P_C vérifie :

1) P_C est strictement positive sur $\mathcal{B}(\Omega)$

2) $\forall \omega \in \Omega, \forall A \subset \Omega, P_C (|\omega| / A) = P_C (|\omega| / A_\omega),$

avec $A_\omega = \{ a \in A ; d(a, \omega) = d(\omega, A) \}$, où $d(\omega, A) = \inf_{a \in A} d(\omega, a)$ et où d est la distance de Manhattan.

Définition 3.8. : Semi-groupe $\{P_t\}$

$\{P_t\}$ ($t \geq 0$) est un semi-groupe, si, pour tout $t \geq 0$, P_t de $P[\Omega] \times P[\Omega]$ dans $[0, 1]$, vérifie les propriétés suivantes :

1) $P_t(A, B) \geq 0$, pour tout $A, B \in P[\Omega]$

2) $\sum_{B \subset \Omega} P_t(A, B) = 1$, pour tout $A \in P[\Omega]$

3) $P_{t+s}(A, B) = \sum_{C \subset \Omega} P_t(A, C) \cdot P_s(C, B)$, $\forall A, B \in P[\Omega]$; $s, t \geq 0$

4) $\lim_{t \rightarrow 0^+} P_t(A, B) = \begin{cases} 1, & \text{si } A = B \\ 0, & \text{si } A \neq B \end{cases}$, pour tout $A, B \in P[\Omega]$

ayant cela, le semi-groupe $\{P_t\}$ a la propriété suivante :

sachant que la configuration est "A" à l'instant 's', la probabilité pour que la configuration soit "B" à l'instant 't+s' est $P_t(A, B)$, cela quelque soit 's'.

Définition 3.9. : Générateur du Semi-groupe $\{P_t\}$

G est un générateur associé au semi-groupe $\{P_t\}$, s'il vérifie les propriétés suivantes :

1) $G(A, B) \geq 0$, pour tout $A \neq B$ dans $P[\Omega]$

2) $\sum_{B \subset \Omega} G(A, B) = 0$, pour tout $A \in P[\Omega]$

3) $P_t(A, B) = e^{tG}(A, B)$, $\forall A, B \in P[\Omega]$ et $\forall t \geq 0$

Définition 3.10. : Processus de contact de Harris

Un processus de Markov sur $P[\Omega]$ est appelé processus de contact de Harris, si son semi-groupe de transition $\{P_t\}$ de naissance et de mort vérifie :

$$\forall A, B \subset \Omega, P_t(A, B) = e^{-t} G(A, B),$$

où G est le générateur associé à $\{P_t\}$, tel que :

- 1) $G(A, B) = 0$, si $|A \Delta B| > 1$
- 2) $G(A, A \cup |\omega|) = \beta(\omega, A)$, si $\omega \notin A$
- 3) $G(A \cup |\omega|, A) = \alpha(\omega, A)$, si $\omega \notin A$
- 4) $G(A, A) = - \sum_{B \neq A} G(A, B)$,

où β est une probabilité de $\Omega \times (P[\Omega] \setminus \{\Omega\})$ vers $]0, 1[$, appelée le taux de naissance et où α est une probabilité de $\Omega \times (P[\Omega] \setminus \{\Omega\})$ vers $]0, 1[$, appelée le taux de mort.

Note : On a utilisé le terme "de naissance et de mort" par analogie aux modèles épidémiologiques (cf. Fricot (85)), i.e. soit A la configuration des sites "malades et sains" ; passer de l'état "sain" à l'état "malade" sera considéré comme étant une "mort" et inversement "naissance".

Définition 3.11. : Le semi-groupe $\{P_t\}$ est dit de renouvellement de naissance et de mort, si son générateur associé vérifie :

- 1) $G(A, A \cup |\omega|) = \beta(\omega, A) = \beta(\omega, A \cup \omega)$, si $\omega \notin A$
- 2) $G(A \cup |\omega|, A) = \alpha(\omega, A) = \alpha(\omega, A \cup \omega)$, si $\omega \notin A$

où $A_\omega = \{a \in A ; d(a, \omega) = d(\omega, A)\}$

Définition 3.12. : Le semi-groupe $\{P_t\}$ est dit Markovien de naissance et de mort, si son générateur associé vérifie :

- 1) $G(A, A \cup |\omega|) = \beta(\omega, A) = \beta(\omega, A \cap N(\omega))$, si $\omega \notin A$
- 2) $G(A \cup |\omega|, A) = \alpha(\omega, A) = \alpha(\omega, A \cap N(\omega))$, si $\omega \notin A$

où $N(\omega)$ est l'ensemble des sites voisins de ω .

Définition 3.13. : Le semi-groupe $\{P_t\}$ de générateur G , irréductible, est dit réversible dans le temps, si pour tout $A, B \in \mathcal{P}[\Omega]$:

$$\mu(A) G(A, B) = \mu(B) G(B, A),$$

où μ est la probabilité stationnaire du processus $\{P_t\}$, ($t \geq 0$), c'est-à-dire vérifie :

$$\sum_{A \subset \Omega} \mu(A) G(A, B) = 0, \quad \forall B \subset \Omega \text{ (cf. Spitzer(74))}$$

II -3.3.2 Convergence

Proposition 3.4. : Soit $\{P_t\}$ un semi-groupe de renouvellement de naissance et de mort, réversible dans le temps, alors : la mesure de Gibbs associée au potentiel U , tel que :

$$\forall A \subset \Omega ; \omega \notin A, e^{U(A \cup \{\omega\}) - U(A)} = \frac{\beta(\omega, A)}{\alpha(\omega, A)},$$

est l'unique mesure invariante pour le processus de contact de Harris défini par β et α (taux de naissance et de mort).

Proposition 3.5. : Le réseau d'automates aléatoires R (cf. § II -3.2.2) associé au potentiel U , réalise le processus de contact de Harris défini par β et α .

Preuves :

1-) pour la démonstration de la proposition 3.4. : voir Fricot(85).

2-) pour la démonstration de la proposition 3.5.: c'est une application directe de la proposition 3.2. et de la proposition 3.4. .

Par conséquent, les deux mesures invariantes du réseau d'automates aléatoires R et celle du processus de contact de Harris ont une expression commune qui est la mesure de Gibbs associée au potentiel U .

II -3.4 Présentation Générale des Réseaux à Seuil

II -3.4.1 Définitions (D'après Fogelman (85) et Robert (86))

Définition 3.14. : Réseau à dynamique synchrone (parallèle)

Soit un réseau d'automates binaires à seuil, i.e. la donnée d'une application F de $\{0,1\}^n$ vers $\{0,1\}^n$, alors on définit le processus discret d'itération parallèle associé au réseau, à partir d'un état initial x^0 , par le système suivant :

$$\forall t \geq 0, x(t+1) = F[x(t)], x(0) = x^0$$

ou encore :

$$\forall i \in \{1, \dots, n\}, \forall t \geq 0, x_i(t+1) = F_i[x_1(t), \dots, x_n(t)], x_i(0) = x_i^0$$

Cela signifie que les automates du réseau changent d'état en parallèle.

Remarque 3.4. : ce type d'itérations correspond à la méthode de Jacobi sur F ou à la méthode des approximations successives en analyse numérique.

Définition 3.15. : Réseau à dynamique séquentielle

Soit F l'application définie ci-dessus, on définit le processus d'itération séquentielle par :

Soit " σ " une fonction de permutation de $\{1, \dots, n\}$ vers $\{1, \dots, n\}$, telle que :

$$\forall t \geq 0, x[t+1] = F_{\sigma}[x(t)], x(0) = x^0, \text{ c'est-à-dire par convention :}$$

$$x_{\sigma^{-1}(1)}(t+1) = F_{\sigma^{-1}(1)}[x_1(t), \dots, x_n(t)]$$

$$x_{\sigma^{-1}(k)}(t+1) = F_{\sigma^{-1}(k)}[x_{\sigma^{-1}(1)}(t+1), \dots, x_{\sigma^{-1}(k-1)}(t+1), x_{\sigma^{-1}(k)}(t), \dots, x_{\sigma^{-1}(n)}(t)],$$

$$\forall k \in \{2, \dots, n\}$$

Cela signifie que les automates du réseau changent d'état l'un après l'autre dans l'ordre donné par la permutation " σ ".

Remarque 3.5. : ce type d'itérations correspond à la méthode de Gauss Seidel dans la résolution des systèmes linéaires.

Définition 3.16. : Réseau à dynamique bloc séquentielle

Soit F l'application définie ci-dessus ; on définit le processus d'itération bloc séquentielle ou partiellement parallèle par :

soit (S_k) , $k = 1, \dots, L$, une partition de l'ensemble des automates du réseau, ordonnée, telle que :

$$\left| \bigcup_{k=1}^L S_k \right| = n \quad (\text{où } |X| : \text{ signifie le cardinal de l'ensemble } X)$$

alors :

$$\forall t \geq 0, x(t+1) = F_{(S_k)} [x(t)], \quad x(0) = x^0 \quad \text{où :}$$

$$\forall i \in S_k, \forall k \in \{1, \dots, L\}, \forall t \geq 0, \quad x_i(t+1) = F_i [z^k(t)] \quad \text{avec } z^k(t) = (z_j^k(t))_j, \text{ tels que :}$$

$$\text{- si } k=1, z^1(t) = (x_1(t), \dots, x_n(t)),$$

$$\text{- si } k \in \{2, \dots, L\}, z_j^k(t) = \begin{cases} x_j(t+1), & \text{si } j \in \bigcup_{i=1}^{k-1} S_i \\ x_j(t), & \text{sinon} \end{cases}$$

Définition 3.17. : Réseau à dynamique stochastique

Soit F l'application définie précédemment ; on définit le processus d'itération aléatoire par :

Pour tout processus stochastique (I_t) , $t \geq 0$, à valeurs dans $\{1, \dots, n\}$, on a :

$$\forall t \geq 0, x(t+1) = F_{(I_t)} [x(t)], \quad x(0) = x^0$$

$$\text{avec} \quad x_i(t+1) = \begin{cases} x_i(t), & \text{si } i \neq I_t(\omega) \\ F_i(x(t)), & \text{si } i = I_t(\omega) \end{cases}$$

c'est-à-dire que le réseau est à dynamique séquentielle avec un choix aléatoire successif des automates "i", où l'automate "i" change d'état et les autres restent inchangés dans l'itération 'i', selon l'ordre donné par le processus aléatoire (I_t) .

Remarques 3.6. :

- Toute itération séquentielle est une itération aléatoire où (I_t) est un processus périodique de période n .
- l'itération chaotiques est un cas particulier des itérations aléatoire (cf. F. Robert (76)).
- Les itérations aléatoires ou chaotiques ont été considérablement utilisées dans les problèmes des verres de spins (cf. J.J. Hopfield(82)).

II -3.4.2 Comportement Dynamique des Réseaux à Seuil

Soit $F : \{0,1\}^n \rightarrow \{0,1\}^n$ un réseau à seuil où les composantes (F_i) , $i = 1, \dots, n$, sont des fonctions à seuil telles que : $\forall i \in \{1, \dots, n\}, \forall x \in \{0,1\}^n$

$$F_i(x_1, \dots, x_n) = \begin{cases} 1, & \text{si } \sum_{j=1}^n a_{ij} x_j \geq \theta_i \\ 0, & \text{si } \sum_{j=1}^n a_{ij} x_j < \theta_i \end{cases}$$

où $A = (a_{ij}) \in \mathbb{R}$ est une matrice carrée $(n \times n)$ et $\theta = (\theta_i) \in \mathbb{R}$ est un vecteur seuil.

Comme $\{0,1\}^n$ est fini, alors le processus $(x(t))$, $t \geq 0$, tel que : $x(t+1) = F[x(t)]$, $t \geq 0$, et $\forall x(0) = x^0 \in \{0,1\}^n$, est périodique de période p , si l'on a :

$$\forall x^0 \in \{0,1\}^n, \exists p, T > 0, \forall t \geq T, x(t+p) = x(t) \quad (1)$$

Soit T l'infimum des temps vérifiant (1).

Par suite, on définit le cycle limite par $\bar{x} = (x^1, \dots, x^p)$ avec $x^1 = x(T)$, \dots , $x^p = x(T+p-1)$, T et p sont appelés respectivement la longueur du transitoire et la période du cycle limite (ou longueur du cycle limite). En effet, pour un cycle limite \bar{x} , défini comme ci-dessus, on définit $S_{\bar{x}}$ l'ensemble des automates (ou les éléments du réseau) stables au cours du cycle limite \bar{x} par :

$$\left\{ \begin{array}{l} S_{\bar{x}} \subset \{1, \dots, n\}, \quad \text{tel que :} \\ \forall i \in S_{\bar{x}}, \forall t \in \{1, \dots, p\}, x_i^t = x_i^1 \end{array} \right.$$

et les automates non-stables (ou oscillants) sont définis par : $\{1, \dots, n\} \setminus S_{\bar{x}}$. Par suite on définit le cœur stable par :

$$SC = \bigcap_{\bar{x}} S_{\bar{x}} \quad (\text{Stable Core})$$

et le cœur oscillant est défini par :

$$OC = \bigcap_{\bar{x}} (\{1, \dots, n\} \setminus S_{\bar{x}})$$

Atlan et al. (81) et Fogelman et al. (82) ont montré que le cœur stable est très robuste, de telle sorte qu'il résiste aux perturbations du réseau ou à l'introduction d'un bruit blanc sur le réseau.

Le procédé le plus puissant pour étudier le comportement dynamique discret consiste à exhiber une fonction 'E' strictement décroissante dite fonction d'énergie, introduite par *J. Hopfield* (cf. *J. Hopfield(82)*), analogue à celle des verres de spins, telle que :

$$E(x^0) > E(x^1) > \dots > E(x^r) = E(x^{r+k}), \quad \forall k \geq 1.$$

En particulier cette méthode a donné des résultats intéressants pour ce qui concerne les réseaux d'automates à seuil.

Posons :

- $E(X) = - (1/2) \cdot X' A X + X' \theta = - (1/2) \cdot \langle AX, X \rangle + \langle \theta, X \rangle, \quad \forall X \in \{0, 1\}^n$
- $2 E(X, Y) = - \langle AY, X \rangle + \langle \theta, X+Y \rangle, \quad \forall X, Y \in \{0, 1\}^n$

où X' est le vecteur transposé de x , où $A = (a_{ij})$ et θ définissent le réseau à seuil et où \langle, \rangle est le produit scalaire classique (ou standard).

Les fonctions 'E' définies ci-dessus sont des fonctions discrètes analogues aux fonctions de Lyapunov, qui sont généralement utilisées pour étudier des systèmes dynamiques continus.

Des chercheurs tels que *Fogelman, Goles, Weisbuch et Pellegrin (83-84)* ont trouvé des résultats intéressants en ce qui concerne la caractérisation du cycle limite (période et longueur du transitoire du cycle) de ces réseaux.

Soit $F(A, \theta)$ le réseau déterministe à seuil où les F_i sont des fonctions à seuil strictes (car on peut toujours se ramener à celui ci) où :

$$F_i (x_1, \dots, x_n) = \begin{cases} 1, & \text{si } \sum_{j=1}^n a_{ij} x_j > \theta_i \\ 0, & \text{si } \sum_{j=1}^n a_{ij} x_j \leq \theta_i \end{cases}$$

et où $A = (a_{ij})$ est la matrice de connexions ou d'interactions carrée ($n \times n$), et $\theta = (\theta_i) \in \mathbb{R}$ est un vecteur seuil.

Citons donc quelques résultats établis selon le mode d'itération utilisé :

1) Cas d'itérations séquentielles

Proposition 3.6. : Soit $F(A, \theta)$ un réseau déterministe à dynamique d'itérations séquentielles ; si $A = (a_{ij}) \in \mathbb{Z}$ est une matrice de connexions carrée symétrique ($n \times n$) dont les éléments diagonaux sont non-négatifs, alors on a :

i) $p = 1$ (seulement des points fixes)

$$\text{ii) } T \leq \sum_{i=1}^n \sum_{j \neq i} |a_{ij}| + 2 \left(\sum_{i=1}^n |a_{ii}| + \sum_{i=1}^n |\theta_i| \right)$$

où p et T sont respectivement la période du cycle limite et la longueur du transitoire de la dynamique.

Remarque 3.7. : Le corollaire suivant est un cas très intéressant dont l'application est par exemple le problème des verres de spins, ...

Corollaire 3.1. : Soit $F(A, \theta)$ un réseau déterministe à dynamique d'itérations séquentielles ; si $A = (a_{ij}) \in \mathbb{Z}$ est une matrice de connexions carrée symétrique ($n \times n$), avec (a_{ij}) dans l'ensemble $\{-1, 0, +1\}$, $\forall i, j \in \{1, \dots, n\}$, et dont les éléments diagonaux sont non-négatifs, alors on a :

$$T \leq 3 \sum_{i=1}^n |V(i)| + 2 \sum_{i=1}^n |a_{ii}| \leq n(3n - 1) \text{ (borne quadratique),}$$

où $|V(i)| = \text{card}\{j \in \{1, \dots, n\} \mid a_{ij} \neq 0, j \neq i\}$.

De plus, si $|V(i)| \leq K$, $\forall i \in \{1, \dots, n\}$ (uniformément en i),

alors : $T \leq (3K + 2)n$ (borne linéaire).

Preuves : (cf. F. Fogelman et al.(83))

2) Cas d'itérations bloc séquentielles

Proposition 3.7. : Soit $F(A, \theta)$ un réseau déterministe à dynamique d'itérations bloc séquentielles ; si $A = (a_{ij}) \in \mathbb{Z}$ est une matrice de connexions carrée symétrique ($n \times n$) et si les $A_k = (a_{ij})$, $i, j \in S_k$, sont non-négatifs, où (S_k) , $k=1, \dots, L$, est une partition ordonnée de l'ensemble $\{1, \dots, n\}$, alors on a :

$$i) \quad T \leq \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| + 2 \sum_{i=1}^n |\theta_i|$$

Corollaire 3.2. : Soit $F(A, \theta)$ un réseau déterministe à dynamique d'itérations bloc séquentielles ; si $A = (a_{ij}) \in \mathbb{Z}$ est une matrice de connexions carrée symétrique ($n \times n$), avec $(a_{ij}) \in \{-1, 0, +1\}$, $\forall i, j \in \{1, \dots, n\}$, et si les $A_k = (a_{ij})$, $i, j \in S_k$, sont non-négatifs, où (S_k) , $k=1, \dots, L$, est une partition ordonnée de l'ensemble $\{1, \dots, n\}$, alors on a :

$$i) \quad p = 1 \quad (\text{la dynamique a seulement des points fixes})$$

$$ii) \quad T \leq 3n^2 \quad (\text{borne quadratique})$$

De plus, si $|V(i)| = \text{card}\{j \in \{1, \dots, n\} \mid a_{ij} \neq 0\} \leq K$, $\forall i \in \{1, \dots, n\}$ (uniformément en i), alors :

$$T \leq (3K)n \quad (\text{borne linéaire})$$

Preuves : (cf. F. Fogelman et al.(84))

3) Cas d'itérations parallèles

Proposition 3.8. : Soit $F(A, \theta)$ un réseau déterministe à dynamique d'itérations parallèles ; et si $A = (a_{ij})$ est une matrice de connexions carrée symétrique ($n \times n$), alors on a :

$$i) \quad p \leq 2$$

si $A = (a_{ij}) \in \mathbb{Z}$ on a de plus : $p = 1$ et

$$ii) \quad T \leq \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| + 2 \sum_{i=1}^n |\theta_i|$$

iii) si $(a_{ij}) \in \{-1, 0, +1\}$, $\forall i, j \in \{1, \dots, n\}$, alors :

$$T \leq 3n^2 \quad (\text{borne quadratique})$$

De plus, si $|V(i)| = \text{card}\{j \in \{1, \dots, n\} \mid a_{ij} \neq 0\} = K$, $\forall i \in \{1, \dots, n\}$, alors :

$$T \leq (3K)n \quad (\text{une borne linéaire})$$

Preuve : (cf. F. Fogelman et al.(84), Goles & Olivos(80)).

Remarque 3.8. : Dans le cas où $A = (a_{ij})$ est seulement symétrique, alors le réseau à seuil $F(A, \theta)$, à dynamique séquentielle, peut avoir des cycles de longueur $p > 1$, i.e. il y en a plus que de points fixes (cf. F. Fogelman et al.(84)).

4) Cas d'itérations aléatoires

Proposition 3.9. : Soit $F(A, \theta)$ un réseau déterministe à dynamique d'itérations aléatoires et si $A = (a_{ij}) \in \mathbb{Z}$ est une matrice de connexions carrée symétrique ($n \times n$) dont les éléments diagonaux sont non-négatifs, et soit (I_t) un processus aléatoire à valeurs dans $\{1, \dots, n\}$, alors l'itération aléatoire associée au réseau $F(A, \theta)$ n'a que des points fixes.

Preuve : D'après la proposition 3.6., il existe une fonction énergie E telle que : E est strictement décroissante, i.e. $x(t+1) \neq x(t) \Rightarrow E[x(t+1)] < E[x(t)]$ où $\forall x \in \{0, 1\}^n$, $E(x) = -(1/2) \sum_{i=1, n} x_i \sum_{j \neq i} a_{ij} x_j + \sum_{i=1, n} (\theta_i - a_{ii}) x_i$; pour la preuve voir Fogelman et al. (83).

Supposons que l'itération admet un cycle de longueur $p > 1$. Soit alors $\bar{x} = (x^1, \dots, x^p)$ le cycle limite de l'itération ; comme l'énergie E est strictement décroissante au cours des itérations, alors : $E(x^1) > \dots > E(x^p) > E(x^1)$ qui est impossible, d'où le résultat.

Remarque 3.9. : Si le processus (I_t) ne visite pas assez souvent certains éléments de $\{1, \dots, n\}$, il peut arriver que le point fixe de l'itération aléatoire ne soit pas le point fixe de l'itération séquentielle.

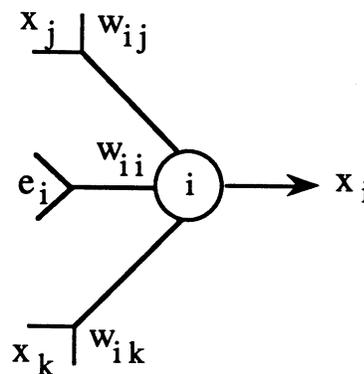
• CAS PARTICULIER :

Il s'agit de réseaux à seuil $F(A, \theta)$ dont la matrice d'interconnexion "A" est non symétrique (quelconque) à valeurs dans $\{-1, 0, +1\}$ (analogue aux verres de spins). Le comportement dynamique de ces réseaux peut être cyclique de longueur grande, d'où l'existence d'un cœur stable et d'un cœur oscillant.

II -4 le modèle neuronal

II -4.1 Introduction

D'après ce qui précède, on choisit Ω (l'ensemble des sites ou neurones de réseau) comme étant une sous-boîte de \mathbb{Z}^2 , telle que $\Omega = [1, \dots, \text{taille}] \times [1, \dots, \text{taille}]$, sur lequel notre réseau est représenté. Le réseau est donc le siège d'une succession de configurations binaires où l'état "1" (resp. "0") représente l'activité (resp. non-activité) du neurone.



$$X_i = f_i(H_i)$$

$$A_i = \sum_{j \in N(i)} W_{ij} X_j + W_{ii} e_i$$

$$H_i = A_i - \theta_i$$

Figure 3 : X_i est l'état de sortie du neurone "i". W_{ij} est le poids de connexion du neurone "j" vers le neurone "i". A_i est une somme pondérée issue des neurones voisins correspondant à l'entrée totale du neurone "i" (appelée fonction d'Harmonie). H_i est une fonction appelée fonction Hamiltonienne. f_i est la fonction de décision qui varie d'un modèle à l'autre : dans notre cas, $f_i(\xi) = \mathbb{1}(\xi)$ où $\mathbb{1}$ est une fonction indicatrice sur $[0, +\infty[$ ou encore $f_i(\xi)$ est respectivement la fonction de Heaviside dans le modèle des réseaux déterministes et la variable aléatoire de loi de Bernoulli $\mathcal{B}[e^\xi / (1 + e^\xi)]$ dans le modèle des réseaux non-déterministes.

Le réseau est donc un système de calcul distribué dont les éléments sont des unités de calcul qui reçoivent des entrées à travers des voisins et ensuite qui produisent leur état de sortie. Autrement dit, le réseau est un ensemble de neurones interconnectés dont chaque neurone "i" est en connexion avec plusieurs neurones. Un de ces neurones est à l'origine de la fibre d'entrée "e_i" (cf. Figure 3) délivrant une information extérieure dite : innovation extérieure et les autres sont les sorties (ou états internes) des neurones voisins

" x_j ", pondérés par des poids synaptiques $W_{ij} \in [-1, +1]^{N(i)}$ (ou poids d'interaction du neurone " j " vers le neurone " i ") qu'on a privilégiés, selon le choix de la topologie (ou structure) de connexion (dans notre cas : sphères de Manhattan centrée en " i ", lieu des points situés à égal distance de " i " pour la norme L_1 , (cf. §IV -2.4 et Figure 7)). Le neurone ensuite calcule son entrée totale dite somme pondérée " A_i " (appelée aussi : *fonction Harmonie*), et il sera actif dès lors que cette somme franchit un certain seuil " θ_i " et inactif dans le cas contraire (cas d'un réseau à seuil déterministe, cf. § II -3.5) ou bien l'état du neurone sera tiré selon une loi de Bernoulli $B[e^{H_i} / (1+e^{H_i})]$, où H_i est une fonction appelée : *fonction Hamiltonienne* (cas d'un réseau non-déterministe ou stochastique, cf. Figure 3).

II -4.2 Présentation du modèle

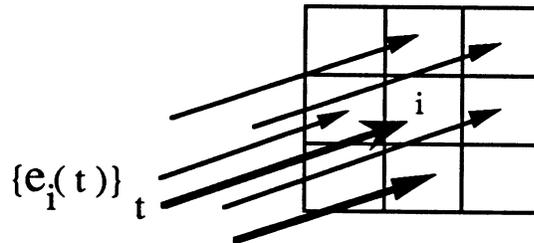


Figure 4 : Réseau à innovation

Soit $x_i(t)$, l'état interne du neurone " i " à l'instant " t " et $e_i(t+1)$, l'innovation à l'instant " $t+1$ " (cf. Figure 4).

Le problème : cherchons une fonction itérative qui fait passer l'état du neurone " i " de $x_i(t)$ à $x_i(t+1)$ en fonction des voisins du neurone " i "

Soit $N(i)$ l'ensemble des voisins (sphères de Manhattan de rayon 1 et 2 en général). On envisage deux cas :

1) - cas déterministe :

Cherchons la fonction F telle que : $X_i(t+1) = F_i[X_j(t), j \in N(i); e_i(t+1)] ; \forall i \in \Omega$

2) - cas stochastique :

Cherchons la probabilité conditionnelle " P " telle que :

$$P = \text{Prob}\{X_i(t+1) = 1 \mid (X_j(t), j \in N(i); e_i(t+1))\}; \forall i \in \Omega$$

- **Calcul de la fonction Harmonie " $A_i(t)$ " :**

$$A_i(t) = \sum_{j \in N(i)} W_{ij}(t) X_j(t) + \dots$$

où W_{ij} sont des coefficients pondérateurs et $X_i(t)$ est l'état interne du neurone " i " à l'instant " t ".

- **Calcul de la fonction Hamiltonienne " $H_i(t)$ " :**

les termes d'ordre supérieur à deux de la fonction $A_i(t)$ sont négligeables parce qu'on se limitera aux configurations singletons et paires. Par suite, on aura la forme de la fonction " H_i " comme suit :

$$H_i(t) = A_i(t-1) + W_{ii}(t-1) e_i(t) - \theta_i(t) ; t \geq 1$$

- **Loi des efficacités synaptiques (ou loi de plasticité) :**

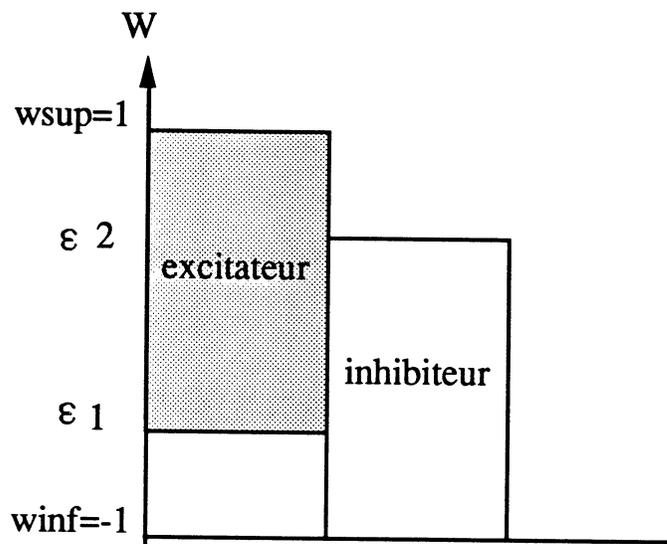


Figure 5 : domaine des poids synaptiques

Nous avons choisi une évolution des efficacités synaptiques, induite par la règle de Hebb (introduite par le physiologiste *Hebb*). Ainsi, on a fixé les valeurs des poids synapses excitatrices (resp. inhibitrices) entre $\epsilon 1$ et $W_{sup}=1$ (resp. entre $W_{inf}=-1$ et $\epsilon 2$), pour des raisons neuro-physiologiques (cf. Figure 5).

• **Interconnectivité :**

Dans un réseau à "n" neurones , il y a $n(n-1)/2$ liaisons deux à deux entre "n" neurones. En privilégiant des liaisons à courte distance, on a arbitrairement utilisé la définition d'un champ radial isotrope à inhibition latérale (cf. Figure 6).

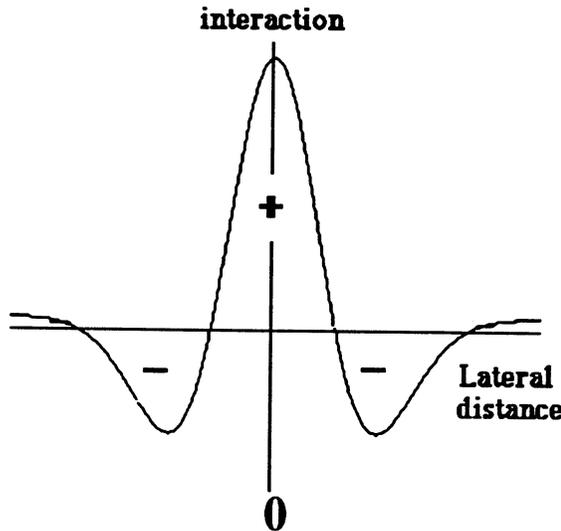


Figure 6 : loi d'interactions spatiale en ' chapeau Mexicain '
 " + " excitation , " - " inhibition

Les interactions s'inspirent de la loi dite en " chapeau mexicain" de sorte que le neurone sera excité par les neurones les plus proches et inhibé par les neurones les plus éloignés. Dans notre cas, le neurone est excité par quatre neurones (sphère de Manhattan de rayon1) et inhibé par huit neurones (sphère de Manhattan de rayon 2) (cf. Figure 7).

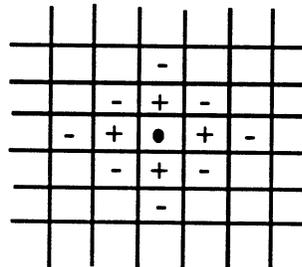


Figure 7 : schéma des connexions inter-neurones
 "+" excitation (manhattan 1)
 "- " inhibition (manhattan 2)

• **La fonction neurone :**

On peut répondre aux questions posées précédemment de la manière suivante : soit 'k' le temps discrétisé tel que $t = k.\Delta t$ et $\theta_i(k)$ le seuil du neurone "i" à l'instant 'k' ; soit $X_j(k-1)$, pour $j \in N(i)$, les états internes des neurones voisins (sphères de Manhattan). On définit donc l'état de sortie du neurone "i" à l'instant 'k' par :

1) **Cas déterministe :**

$$\begin{cases} X_i(k) = \mathbb{1}\left(\frac{e^{H_i(k)}}{1+e^{H_i(k)}} - \frac{1}{2}\right) \\ \text{où } \mathbb{1}(x) = \begin{cases} 1, & \text{si } x > 0 \\ 0, & \text{si } x \leq 0 \end{cases} \end{cases}$$

2) **Cas stochastique :**

$$P = \text{Prob} \left\{ X_i(k) = 1 \mid X_j(k-1) ; j \in N(i), e_i(k) \right\} = \frac{e^{H_i(k)}}{1+e^{H_i(k)}}$$

où H_i est la fonction Hamiltonienne associée au réseau (cf. pp. 36).

Note : pour plus de détails voir chapitre IV : *Implantation et Réalisation*.

• **L'algorithme du Réseau :**

A partir d'un état initial du réseau : X^0 , on s'intéresse au processus d'évolution du type Hebbien suivant :

$$X(k+1) = F[X(k) ; e(k+1)] , \quad \forall k \geq 0, X(0) = X^0 ,$$

où F est déterministe ou non-déterministe, selon le cas des réseaux traités.

Définition 4.1. : *Un réseau d'apprentissage du type Hebbien appelé apprentissage non-dirigé, est la donnée d'une famille $\{\Psi_X\}_{X \in \{0,1\}^\Omega}$ de fonctions, telles que :*

$$\Psi_X : [-1,+1]^{|\Omega|^2} \longrightarrow [-1,+1]^{|\Omega|^2}$$

$$W \quad \mapsto \Psi_X(W),$$

où $\Psi_X(W) = \{\Psi_{x_i, x_j}(w_{ij})\}_{i, j \in \Omega}$ telles que :

$$\Psi_{0,0}(w_{ij}) = w_{ij}$$

$$\Psi_{0,1}(w_{ij}) = \Psi_{1,0}(w_{ij}) \text{ est une contraction sur } [-1,+1]$$

$$\Psi_{1,1}(w_{ij}) = \text{est une dilatation sur } [-1,+1]$$

Remarque 4.1.: Dans nos simulations (cf. chapitre IV), on a pris Ψ_X comme suit :

$$\Psi_{x_i^k, x_j^k}(w_{ij}(k)) = w_{ij}(k) + \alpha \cdot f(w_{ij}(k)) \cdot \lambda(x_i(k), x_j(k)),$$

avec $x_j(k) = e_i(k+1)$, si $j = i$, où

$$\begin{cases} \lambda(0,0) = 0 \\ \lambda(0,1) = \lambda(1,0) = a \\ \lambda(1,1) = b \end{cases}$$

et où a et b sont des constantes bien choisies (cf. chapitre IV)

Enfin, le système d'évolution du réseau s'écrit de la manière suivante :

$$\begin{cases} x_i^{k+1} = F(H_i(k)); i \in \Omega \\ W_{ij}(k+1) = \Psi_{x_i^k, x_j^k}(W_{ij}(k)); \text{ si } j = i, x_j^k = e_i^{k+1}; i, j \in \Omega \\ \theta_i(k+1) = \varphi(\theta_i(k) - \alpha A_i(k)); i \in \Omega \end{cases}$$

où Ψ_X est la fonction du type *Hebb*, définie ci-dessus, et φ est une fonction bornée de \mathbb{R} dans \mathbb{R} .

II -4.3. Convergence et Mesures Invariantes (d'après O. François (90))

• **Rappels :**

On a vu, dans le cas des réseaux déterministes à seuil (cf. § II -3.5), que le comportement dynamique, dans différents modes d'itérations, converge vers des états (ou des configurations) stables (des points fixes) ou vers des cycles limites (comportement périodique), après un certain temps (appelé temps transitoire). Parfois, le système d'évolution peut avoir un comportement "chaotique", c'est-à-dire que le système explore successivement un très grand nombre de configurations possibles.

Si la taille du réseau est $|\Omega|=n$, alors on a a priori 2^n configurations possible à observer, ce qui représente une grande variété de comportements possibles différents. En réalité, le biologiste S.A. Kauffman, et beaucoup d'autres, ont observé par des simulations sur ordinateur que le réseau aléatoire binaire, d'interconnectivité d'ordre 2 itéré en parallèle, adopte un comportement périodique, après une période transitoire, de durée faible. Ainsi, les simulations montrent que :

- 1)- la période du cycle est en moyenne de l'ordre de \sqrt{n} , ce qui représente une réduction énorme par rapport aux 2^n configurations possibles a priori.
- 2)- le nombre de cycles limites différents est en moyenne égal à $\sqrt{n} / 2$, ce qui représente un très petit nombre de comportements possibles par rapport à la taille du réseau .

Notes :

- On suppose que $W = (w_{ij})$ est une matrice de connexions régulières (invariantes par translation), $\theta^k = \theta$ et que les tirages des états des neurones sont indépendants.

- On note : $\{i\} = (0_{1..}, I_i, ..0_n)$,

$$H_i(x) = \sum_{j \in \Omega} w_{ij} x_j^k - \theta_i = \langle Wx, \{i\} \rangle - \langle \theta, \{i\} \rangle ,$$

où \langle , \rangle désigne le produit scalaire classique (ou standard).

En ce qui concerne les réseaux non-déterministes, on a vu que (cf. § II -3.3) le processus d'itération purement séquentielle admet une unique mesure invariante, qui s'installe après un régime transitoire ; c'est la mesure de Gibbs associée à un certain potentiel U , tel que :

$$\mu_S(x) = \frac{e^{U(x)}}{Z}$$

où $U(x) = \langle Wx, x \rangle - \langle \theta, x \rangle$ et où Z est la constante de normalisation.

Dans la suite nous allons traiter le cas d'un processus d'itération totalement parallèle et partiellement parallèle (ou bloc-séquentielle), pour les réseaux, simples, totalement connectés (cf. pp. 40).

La règle du changement d'état pour chaque neurone est choisie non-déterministe (appelée aussi : *Règle de Transition Locale* - notée par : "RTL") : c'est la probabilité pour que le neurone "i" soit dans l'état '1' à l'instant 'k+1' sachant l'état du réseau à l'instant 'k', notée :

$$\text{Prob}\{X_i(k+1) = 1 \mid X(k) = x\} = \frac{e^{(\sum_{j \in \Omega} w_{ij} x_j^k - \theta_i)}}{1 + e^{(\sum_{j \in \Omega} w_{ij} x_j^k - \theta_i)}}$$

D'une manière générale : le neurone "i" est soumis à la règle de transition locale si :

$$\text{Prob}\{X_i(k+1) = y_i \mid X(k) = x\} = \frac{e^{y_i (\sum_{j \in \Omega} w_{ij} x_j^k - \theta_i)}}{1 + e^{(\sum_{j \in \Omega} w_{ij} x_j^k - \theta_i)}}, \forall y_i \in \{0, 1\}$$

où $W = (w_{ij})$ est la matrice des poids synaptique et $\theta = (\theta_i)$ est le vecteur seuil.

• **Cas d'un processus d'itération totalement parallèle :**

Le processus $\{X(t)\}_{t \geq 0}$, correspondant au réseau non-déterministe est un processus de Markov homogène ; soit M sa matrice de transition.

Proposition 4.1. : *Le processus d'itération $\{X(t)\}$ totalement parallèle est ergodique.*

Preuve : Soit M la matrice de transition du processus $\{X(t)\}$; puisque les tirages sont indépendants, alors, pour tout $x, y \in \{0, 1\}^\Omega$, on a :

$$M(x, y) = \text{Prob}\{X(k+1) = y \mid X(k) = x\}$$

$$\Rightarrow M(x, y) = \text{Prob}\left\{ \bigcap_{i \in \Omega} X_i(k+1) = y_i \mid X(k) = x \right\}$$

$$\Rightarrow M(x, y) = \prod_{i \in \Omega} \text{Prob}\{X_i(k+1) = y_i \mid X(k) = x\}$$

$$d'o\grave{u} \quad M(x, y) = \frac{e^{\langle Wx, y \rangle - \langle \theta, y \rangle}}{\prod_{i \in \Omega} (1 + e^{\langle Wx, \{i\} \rangle - \langle \theta, \{i\} \rangle})} > 0, \quad \forall x, y \in \{0, 1\}^\Omega$$

En vertu du th eor eme de *Perron-Frobenius*, le processus $\{X(k)\}$ est ergodique.

Proposition 4.2. : *Supposons que la matrice de connexions $W = (w_{ij})$ est sym etrique et que $\theta = 0$, alors la mesure de Gibbs μ associ ee au potentiel U est invariante pour le processus d'it eration totalement parall ele, et est telle que, en posant $\Gamma = \{0, 1\}^\Omega$, l'on ait :*

$$\mu_p(x) = \frac{\sum_{y \in \Gamma} e^{\langle Wx, y \rangle}}{Z}, \quad \text{o\grave{u } } Z \text{ est la constante de normalisation,}$$

$$\text{et o\grave{u } } U(x) = \sum_{i \in \Omega} \log(1 + e^{\langle Wx, \{i\} \rangle})$$

Preuve: D'apr es la proposition 4.1., la matrice de transition M du processus $\{X(k)\}$ s' ecrit de mani ere g en erale, pour un seuil θ non nul :

$$M(x, y) = \frac{e^{\langle Wx, y \rangle - \langle \theta, y \rangle}}{d(x, \theta)}, \quad \text{o\grave{u } } d(x, \theta) = \prod_{i \in \Omega} (1 + e^{\langle Wx, \{i\} \rangle - \langle \theta, \{i\} \rangle})$$

En effet, du fait que $M(x, \cdot)$ est une probabilit e, alors $d(x, \theta)$ peut s' ecrire aussi comme suit :

$$d(x, \theta) = \sum_{S \in \Gamma} e^{\langle Wx, S \rangle - \langle \theta, S \rangle} = \prod_{i \in \Omega} (1 + e^{\langle Wx, \{i\} \rangle - \langle \theta, \{i\} \rangle})$$

$$\Rightarrow \sum_{S \in \Gamma} e^{\langle Wx, S \rangle - \langle \theta, S \rangle} \cdot M(x, y) = e^{\langle Wx, y \rangle - \langle \theta, y \rangle}$$

$$\Rightarrow \sum_{X \in \Gamma} \sum_{S \in \Gamma} e^{\langle Wx, S \rangle - \langle \theta, S \rangle} \cdot M(x, y) = \sum_{X \in \Gamma} e^{\langle Wx, y \rangle - \langle \theta, y \rangle} \quad (*)$$

• Si on suppose que le seuil est nul, i.e. $\theta = 0$, alors l'équation (*) peut s'écrire comme suit :

$$\sum_{X \in \Gamma} \sum_{S \in \Gamma} e^{\langle Wx, S \rangle} \cdot M(x, y) = \sum_{X \in \Gamma} e^{\langle Wx, y \rangle}$$

$$\Rightarrow \sum_{X \in \Gamma} \sum_{S \in \Gamma} e^{\langle Wx, S \rangle} \cdot M(x, y) = \sum_{X \in \Gamma} e^{\langle Wy, x \rangle} \quad (\text{car } W \text{ est symétrique})$$

$$\Rightarrow \sum_{X \in \Gamma} \mu_p(x) \cdot M(x, y) = \mu_p(y)$$

d'où μ_p est invariante par M .

Remarque 4.3. : On peut se ramener dans la proposition 4.2. au cas où le vecteur seuil θ sera non nul de la manière suivante :

avec :

$$\forall \bar{x} \in \Gamma^* = \{0, 1\}^{[1..n+1]}, H_i(\bar{x}) = \sum_j \bar{w}_{ij} \bar{x}_j = \langle \bar{W} \bar{x}, \{i\} \rangle$$

$$\bar{W} = \begin{pmatrix} W & -\theta \\ -\theta & 0 \end{pmatrix}$$

$${}^t \bar{x} = (x_1, \dots, x_i, \dots, x_n, 1), \quad \text{et } {}^t \{i\} = (0_1, \dots, 1_i, \dots, 0_{n+1}).$$

• **Cas d'un processus d'itérations partiellement parallèles :**

D'après Fogelman (85), soit Ω une sous-boîte de \mathbb{Z}^2 constituant le réseau de neurones et soit $\{\Omega_i\}, i=1, \dots, N$, une partition de Ω dont chaque Ω_i contient des éléments non deux à deux connectés, c'est à dire : $\forall k, j \in \Omega_i, w_{kj} = 0$.

On définit alors le processus $\{X(t)\}$ d'itération partiellement parallèle par :

$$\forall t \geq 1, X(t) = F_{\{\Omega_i\}}(X(t-1)), X(0) = X^0,$$

où X^0 est l'état initiale du réseau et où :

$$\forall t \geq 1, \forall i \in \Omega, X_i(t) = \begin{cases} x_i(t-1), & \text{si } i \notin \Omega_t \text{ mod}(N) \\ \text{tiré selon la règle "RTL",} & \text{si } i \in \Omega_t \text{ mod}(N) \end{cases}$$

Remarque 4.4.: Le processus $\{X(t)\}$ d'itérations partiellement parallèles est un processus de Markov non-homogène. Par contre, le sous processus $\{X(tN)\}$ est un processus de Markov homogène.

Proposition 4.3.: Le processus $\{X(tN)\}_{k \in \mathbb{N}}$ d'itérations partiellement parallèle est ergodique.

Preuve: Soit M la matrice de transition du processus $\{X(tN)\}$; puisque les tirages sont supposés indépendants, alors on peut écrire M sous la forme suivante :

$$M = \prod_{i=1}^N M_i$$

où M_i est la matrice de transition de l'étape ti à l'étape $(t+1)i$ donnée par :

$$M_i(x, y) = \begin{cases} 0, & \text{si } x \neq y \text{ sur } \Omega \setminus \Omega_i \\ \prod_{k \in \Omega_i} \frac{e^{y_k (\sum_{j \in \Omega} w_{kj} x_j^t - \theta_k)}}{1 + e^{(\sum_{j \in \Omega} w_{kj} x_j^t - \theta_k)}}, & \text{sinon} \end{cases}$$

Or, $M_i(x, y)$ peut s'écrire aussi comme : $\forall x, y \in \{0,1\}^\Omega$

$$M_i(x, y) = \begin{cases} 0, & \text{si } x \neq y \text{ sur } \Omega \setminus \Omega_i \\ \frac{e^{\langle Wx, y^i \rangle - \langle \theta, y^i \rangle}}{d_i(x, \theta)}, & \text{sinon} \end{cases}$$

où $y^i = (y_{i,k})_{k \in \Omega}$, avec $y_{i,k} = y_k$, si $k \in \Omega_i$;

$$y_{i,k} = 0, \quad \text{sinon}$$

et où

$$d_i(x, \theta) = \prod_{k \in \Omega_i} (1 + e^{\langle Wx, \{k\} \rangle - \langle \theta, \{k\} \rangle})$$

Par conséquent :

$$\forall x, y \in \{0,1\}^\Omega, \quad M(x, y) = \prod_{i=1}^N M_i(E_i, E_{i-1}), \text{ où}$$

$$E_i = (X - \tilde{X}^i) \cup \tilde{Y}^i, \text{ avec } \begin{cases} \tilde{X}^i = \bigcup_{j \leq i} (X \cap \Omega_j), \tilde{X}^0 = 0 \\ \tilde{Y}^i = \bigcup_{j \leq i} (Y \cap \Omega_j), \tilde{Y}^0 = 0 \end{cases}$$

Comme M est irréductible, c'est-à-dire $\forall x, y \in \{0,1\}^\Omega M(x, y) > 0$, d'après le théorème de Perron-Frobenius, le processus $\{X(tN)\}$ est ergodique.

Proposition 4.4. : Si $\forall i=1..N, \forall k, j \in \Omega_i, w_{kj} = 0$ (i.e. les éléments de Ω_i sont non deux à deux connectés), alors la mesure de Gibbs μ est l'unique mesure invariante pour le processus d'itération partiellement parallèle, telle que :

$$\mu_S(x) = \frac{e^{\langle Wx, x \rangle - \langle \theta, x \rangle}}{Z}, \text{ où } Z \text{ est la constante de normalisation.}$$

Preuve : D'après la proposition 4.3., il existe une unique mesure μ , telle que : $\mu = \mu M$ (en vertu du théorème de Perron-Frobenius).

Il reste maintenant à montrer que la mesure de Gibbs $\mu(x) = \frac{e^{U(x)}}{Z}$, associée au potentiel U : $U(x) = \langle Wx, x \rangle - \langle \theta, x \rangle$, est invariante à gauche par la matrice M ; en fait, elle l'est pour chaque $M_i, i=1, \dots, N$, c'est-à-dire :

$$\mu(x) = \sum_{y \in \Omega} \mu(y) M_i(y, x)$$

En posant $\Gamma = \{0,1\}^\Omega$ et $\Gamma_i = \{0,1\}^{\Omega_i}$, nous avons :

- Soit $x \in \Gamma$, tel que $x = 0$ sur Ω_i

$$\mu(x) = \sum_{y^i \in \Gamma^i} \mu(x \cup y^i) M_i(x \cup y^i, x)$$

$$\Rightarrow \mu(x) = \sum_{y^i \in \Gamma^i} \frac{\mu(x \cup y^i)}{d_i(x \cup y^i, \theta)}$$

$$\Rightarrow \mu(x) = \sum_{y^i \in \Gamma^i} \frac{\mu(x \cup y^i)}{d_i(x, \theta)} \quad (\text{car } w_{k,j} = 0, \forall k, j \in \Omega_i)$$

Soit ensuite $u^i \in \Gamma_i$,

$$\mu(x \cup u^i) = \sum_{y^i \in \Gamma^i} \mu(x \cup y^i) M_i(x \cup y^i, x \cup u^i)$$

$$\Rightarrow \mu(x \cup u^i) = \sum_{y^i \in \Gamma^i} \mu(x \cup y^i) \frac{e^{\langle W(x \cup y^i), u^i \rangle - \langle \theta, u^i \rangle}}{d_i(x \cup y^i, \theta)}$$

$$\Rightarrow \mu(x \cup u^i) = \sum_{y^i \in \Gamma^i} \mu(x \cup y^i) \frac{e^{\langle Wx, u^i \rangle - \langle \theta, u^i \rangle}}{d_i(x, \theta)} \quad (\text{car } w_{k,j} = 0, \forall k, j \in \Omega_i)$$

$$\Rightarrow \mu(x \cup u^i) = \mu(x) e^{\langle Wx, u^i \rangle - \langle \theta, u^i \rangle} \quad (\oplus)$$

- De même, si $x = s^i \neq 0$ sur Ω_i , si on pose $z = x - s^i$ sur Ω_i , alors on revient au cas précédent, i.e. $z = 0$ sur Ω_i .

Par suite, on vérifie aisément que $\mu(x) = \frac{e^{\langle Wx, x \rangle - \langle \theta, x \rangle}}{Z}$ est solution du système d'équations donné par la relation (\oplus) , d'où $\mu = \mu_S$ est invariante par M .

Remarque 4.5. : On remarque que les deux mesures invariantes μ_S et μ_P peuvent être très différentes pour certaines valeurs de la matrice de connexions $W = (w_{ij})$; par exemple on a :

$$\lim_{w_{11} \rightarrow -\infty} \delta(\mu_P, \mu_S) = +\infty, \text{ où } \delta \text{ est la distance de Kullback.}$$

II -4.4 Estimation de la Mesure Gibbs

Notation : On note la configuration nulle par " \emptyset ".

On dit que la configuration $A \in \{0, 1\}^{\Omega}$ est singleton (ou configuration *1-ton*) si elle possède un seul neurone excité "1", paire (ou configuration *2-ton*) si elle possède deux neurones excités "1", triplet (ou configuration *3-ton*) si elle possède trois neurones excités "1", etc...

Dans un réseau à $|\Omega| = N$ neurones, il y a 2^N configurations possibles telles que :
 N configurations possibles sont des *1-ton* (singletons), C_N^2 configurations possibles sont des *2-ton* (paires), C_N^3 configurations possibles sont des *3-ton* (triplets), ..., C_N^k configurations possibles sont des *k-ton*.

EXEMPLE : Si $N = 32 \times 32$

Il y a $2^{(32)^2}$ configurations possibles :
 $N = 1024$ configurations de *1-ton* (singleton),
 $C_N^2 = 523776$ configurations de *2-ton* (paires),
 etc...

• Calcul des potentiels d'interactions

- Si $A = \{i\}$ (configuration singleton)

$$J_U[\{i\}] = \log \left[\frac{\mu(\{i\})}{\mu(\emptyset)} \right]$$

- Si $A = \{i, j\}$ (configuration paire)

$$J_U[\{i, j\}] = -\log \left[\frac{\mu(\{i\})}{\mu(\emptyset)} \right] - \log \left[\frac{\mu(\{j\})}{\mu(\emptyset)} \right] + \log \left[\frac{\mu(\{i, j\})}{\mu(\emptyset)} \right]$$

Pour estimer la mesure de Gibbs associée à la dynamique du réseau, il faut mesurer (ou estimer) tous les potentiels d'interactions du réseau de $N = \text{taille}^2$ neurones qui correspondent au total à 2^N configurations potentielles. Or, d'après ce qui précède, il est impossible de calculer tout ces potentiels, car on serait submergé par les calculs. Pour éviter tout cela, il nous suffit de ne laisser apparaître que des configurations singletons, paires et triplets. Pour cela il, suffit de choisir une bonne valeur du paramètre " Δt " (le pas d'échantillonnage de la dynamique du réseau) suffisamment petit (cf. §IV -2.9).

Dans la pratique (cf. § IV -3.2.3, § IV -3.2.4), on estime les potentiels d'interaction durant un certain temps " T " assez long, où $T = (\text{iter_max}) \cdot \Delta t$, en remplaçant la mesure μ dans les relations précédentes par sa mesure empirique (ou fréquence empirique) μ^* .

On calcule donc J_U^* et U^* par :

$$U^*(A) = \log [\mu^*(A)] - \log [\mu^*(\emptyset)]$$

$$\Rightarrow U^*(A) = \sum_{B \subset A} J_U^*(B), \text{ avec } J_U^*(\emptyset) = 0.$$

Et ensuite, on applique la formule d'inversion classique de Mœbius :

$$J_U^*(A) = \sum_{B \subset A} (-1)^{|A \setminus B|} U^*(B)$$

où $|A \setminus B|$ est le nombre de " I " qui sont dans la configuration " A " et qui ne sont pas dans " B " (cf. B. D. Ripley (88)).

CHAPITRE III

DESCRIPTION DE LA MACHINE PARALLELE UTILISEE : "L'HYPERCUBE [FPS] T-40"

III -1. Introduction

Depuis Von Neumann, les ordinateurs ont évolué dans le sens d'une augmentation des performances de leurs divers composants (contrôle, traitement, mémoire). Les machines séquentielles arrivent maintenant près des limites physiques de leur rapidité. L'introduction du parallélisme devient alors nécessaire pour accélérer les traitements.

Les fabricants de super-calculateurs s'intéressent ainsi à cette nouvelle idée de parallélisme, et proposent des performances à un coût inférieur à celui des meilleures machines séquentielles du moment. Parmi les super-calculateurs disponibles sur le marché, apparaissent ainsi des machines multiprocesseurs. L'hypercube [FPS] T-40 (cf. Figure 1) fait partie de ces machines et possède un parallélisme massif (virtuel car la machine est extensible de $8=2^3$ jusqu'à $16384=2^{14}$ processeurs) entre les processeurs.

Dans ce chapitre, on présente une introduction globale sur le parallélisme et notamment sur le type d'ordinateur utilisé le long de cette thèse : l'hypercube [FPS] T-40 (à 32 processeurs) couplé à un *Micro-Vax* dont le système d'exploitation est UNIX.

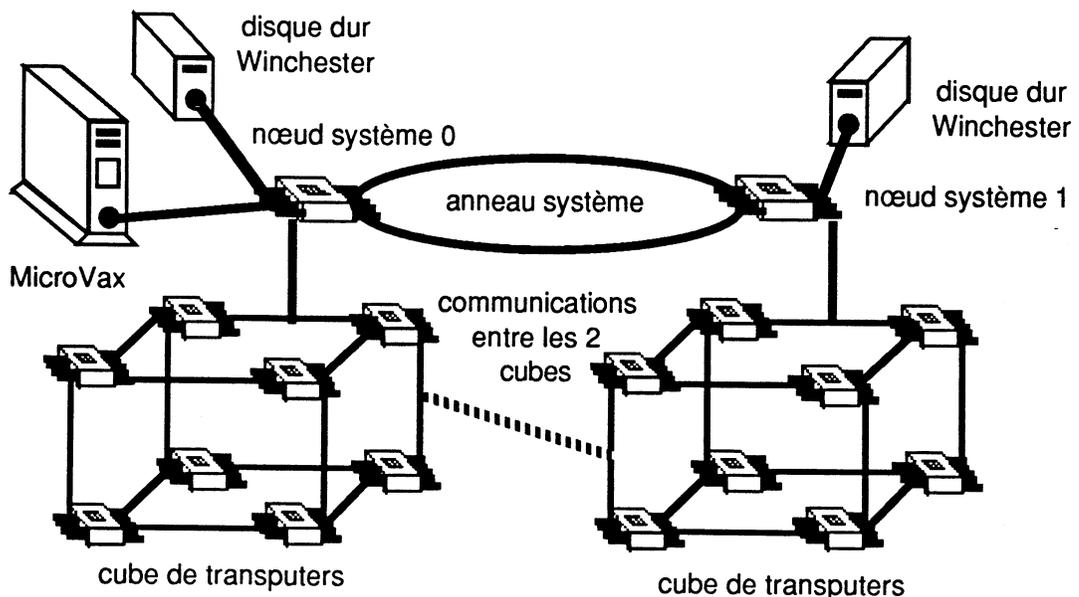


Figure 1 : Organisation générale du T - 20

III -2. Une vue globale sur le parallélisme

III -2.1 Introduction au parallélisme

Au fur et à mesure que nous affinons les modèles mathématiques, les calculs que nous aurions besoin d'effectuer sont de plus en plus complexes. En outre, les ordinateurs scientifiques réclament de plus en plus de performances. En effet, le calcul scientifique demande des ordinateurs de plus en plus puissants pour améliorer le traitement de modèles proches de la réalité que ce soit pour étudier le comportement aérodynamique d'un avion, la fission ou la fusion de particules nucléaires, l'évolution d'un cyclone, l'interprétation d'images ou de la voix humaine, etc... Ces opérations sont de plus en plus coûteuses.

Si l'on prend comme base le million d'opérations flottantes par secondes (le mégaflop), on obtient les chiffres suivants : *100* pour l'aérodynamique, plus de *200* pour les études sur les plasmas, *500* pour la conception d'une centrale nucléaire, très supérieur à *1000* pour les prévisions météorologiques. Or, pour mener des calculs de cette taille à bien, même pour les plus puissants des ordinateurs actuels, il faut un temps relativement long : de quelques heures à plusieurs semaines, suivant les cas.

Si l'on regarde la plupart des calculs qui sont effectués par les ordinateurs monoprocesseurs, on constate qu'une part très importante du temps machine est destiné à un petit nombre de tâches bien spécifiques telles que : transformation de Fourier, produit ou inversion de matrices, résolution des systèmes linéaires, etc... ; si on pouvait les accélérer, on gagnerait beaucoup. Cependant, ces dernières années, sont apparues des machines basées sur les réalisations de circuits intégrés qui ont pu gagner un temps énorme. Malheureusement, ces techniques se rapprochent de leur limites ; il faut donc envisager de nouvelles techniques. L'une d'entre elles est la technique des super-calculateurs parallèles tels que : *HYPERCUBE (T-série, [FPS], n-cube)*, *T-Node*, *MasPar*, *CONNECTION MACHINE*, etc... Même si on ne peut plus espérer une amélioration au niveau des performances d'un seul processeur, on pourra gagner du temps si on distribue une tâche de calcul à plusieurs processeurs travaillant en parallèle. Mais ces machines sont encore des calculateurs non spécialisés.

Certains chercheurs (*de l'université de Carnegie Mellon, U.S.A*) (cf. *Kung(82) et Leiserson(83)*), (cf. *Quinton & al.(89)*), ont développé la notion des réseaux, dits réseaux systoliques, qui ont pour objectif de rendre ces calculateurs plus spécifiques. Ces réseaux sont des réseaux d'automates cellulaires réalisés par des circuits *VLSI* (*Very Large Scale Integrated Circuit*). On peut envisager une structure de calcul très rapide à partir d'un ordinateur hôte, qui relie des réseaux chargés des calculs spécifiques les plus coûteux, implémentés en *VLSI*.

Pour répondre à ces besoins, les constructeurs s'appuyaient traditionnellement sur les progrès des circuits intégrés. Cependant, la technologie *VLSI* actuelle, avec des circuits intégrés au silicium tend à arriver à ses limites : les niveaux d'intégration sont très élevés et les connexions entre circuits sont raccourcies au maximum. On pourra sans aucun doute encore gagner de la vitesse avec cette technologie, mais sans aucun rapport avec les gains du passé.

Une première solution envisageable peut être de changer de technologie, c'est-à-dire d'utiliser de nouveaux matériaux semi-conducteurs comme l'arséniure de Gallium (*AsGa*), des transistors à base métallique ou encore des matériaux supra-conducteurs. Mais à l'heure actuelle, ces techniques restent expérimentales et de plus sont très chères par rapport aux technologies traditionnelles du silicium.

Une seconde approche, en vue d'augmenter les performances d'un ordinateur, consiste à bouleverser sa composition classique. En effet, le modèle cité par *John VON NEUMANN en 1945* décrit un ordinateur comme une machine éminemment séquentielle : les instructions sont extraites de la mémoire, décodées, puis exécutées, puis les résultats sont rangés en mémoire avant de passer à l'instruction suivante. Il peut donc être fort intéressant d'introduire un certain parallélisme dans la structure des ordinateurs, tant au niveau de l'architecture globale par l'utilisation de plusieurs processeurs qu'au niveau du processeur local. Les algorithmes et les méthodes qui seront utilisés sur ce type d'ordinateurs seront donc d'une conception nouvelle, puisqu'ils manipulent de nouveaux concepts. On pourra se permettre d'exécuter simultanément plusieurs instructions et donc d'accélérer notablement les calculs.

Une description plus complète des divers composants d'un ordinateur classique pourra permettre une meilleure compréhension des bouleversement apportés par les machines parallèles. La machine de *John VON NEUMANN*, encore appelée machine *SISD* (*Single Instruction Single Data*), correspond à un seul flot d'instructions et un seul

flot de données ; elle comprend un processeur (l'unité de traitement) et une mémoire (cf. *Figure 2*). Celle-ci est organisée en "cellules", accessibles individuellement via un nombre (l'adresse de la cellule), qui contiennent les instructions du programme et les données nécessaires à son exécution.

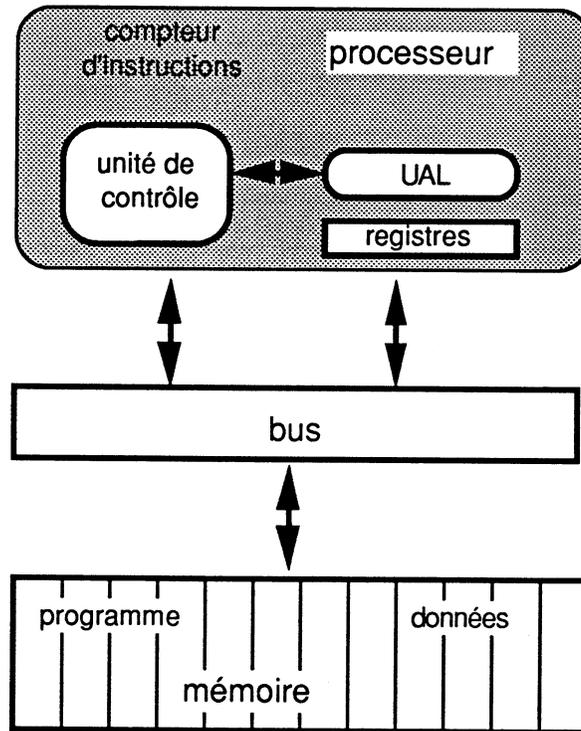


Figure 2 : ordinateur séquentiel

Le **processeur**, qui constitue la partie active de la machine, est constitué de quatre unités principales :

- *Le compteur d'instructions (ou compteur ordinal),*
- *L'unité de contrôle (encore appelée unité de commande),*
- *L'unité arithmétique et logique et les registres.*

L'unité de contrôle, véritable chef d'orchestre de l'ordinateur, consulte le compteur d'instructions pour lire en mémoire la nouvelle instruction qu'il décode et exécute. Suivant le type d'instruction, une donnée peut être lue ou écrite en mémoire ou une opération effectuée.

Les registres sont de petites unités de mémoire à accès très rapide où sont généralement stockées les données les plus utilisées.

L'unité arithmétique et logique (*UAL*) exécute les calculs ; elle est généralement munie de circuits câblés qui lui permettent de réaliser de façon matérielle les principales opérations arithmétiques (addition, soustraction, multiplication et division).

Le *bus* permet une gestion efficace des communications entre les diverses parties de l'ordinateur :

- *Processeur*,
- *mémoire* ,
- *entrées / sorties*.

L'idée qui préside au parallélisme est simple : si diverses parties d'un algorithme sont indépendantes et que l'on dispose de plusieurs unités de traitement, alors elles peuvent être effectuées simultanément sur les diverses unités. Il faut cependant s'assurer que les dites parties sont bien indépendantes et qu'elles n'auront pas besoin d'avoir simultanément accès aux mêmes données.

Il est bien évident qu'il existe de nombreux styles d'architectures parallèles : ceux-ci dépendent de la façon dont les instructions sont exécutées, des topologies de communication des processeurs, et de la structure de la mémoire (*cf. Quinton (85)*).

Une des classifications les plus communes décompose le parallélisme en trois classes, suivant le mode de contrôle des instructions élémentaires, en distinguant les flots de données et les flots d'instructions. On a alors les architectures *pipeline*, *SIMD* et *MIMD*.

Les fabricants de super-calculateurs s'intéressent à ces nouvelles idées et les mettent en application en proposant sur le marché des ordinateurs parallèles aux performances élevées à un coût inférieur à celui des meilleures machines séquentielles du moment.

Le constructeur [*FPS*] (*Floating Point System*) se situe dans cette lignée en proposant plusieurs machines multiprocesseurs (les T-series machines) dont le *T-40*, [*FPS*] (*cf. Figure 3*) sur lequel on a implanté notre algorithme.

III -3. Présentation du T-40 [FPS]

En raison de leurs performances et de leur capacité de mémoire, les ordinateurs parallèles de type hypercube deviennent de sérieux concurrents dans la course aux Mégaflops, pour les applications scientifiques (cf. Figure 1, Figure 3).

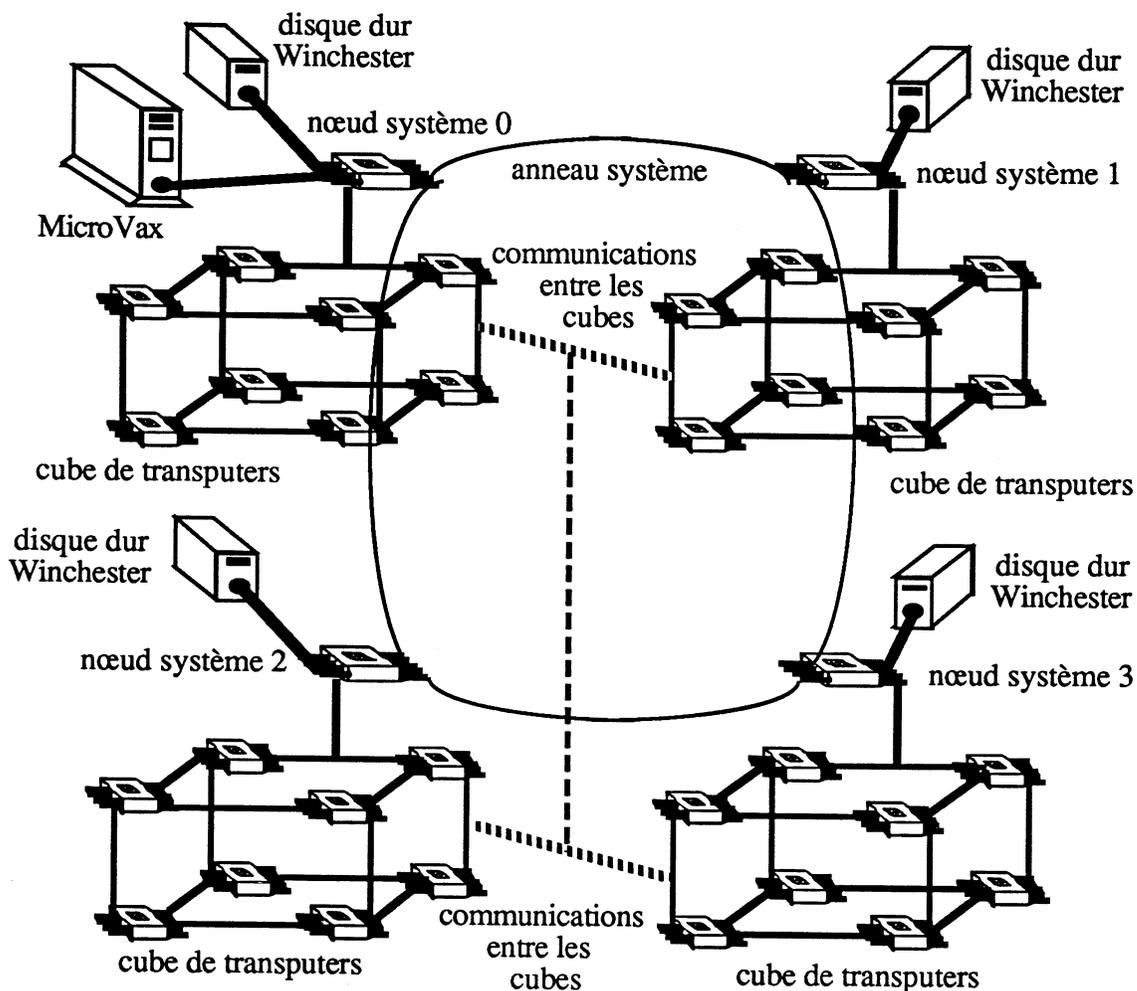


Figure 3 : Le T-40, une machine parallèle à mémoire distribuée

III -3.1 Généralités

Les *T-series* machines exploitent à fond une nouvelle architecture de machines parallèles, de sorte que ses composantes (ou processeurs) sont arrangées d'une manière optimale dans un hypercube.

Une *T-series* machine peut être considérée comme une architecture n -cube $=\{0,1\}^n$ où chaque sommet correspond à un processeur. Il y a au total $N = 2^n$ processeurs dans une *T-series* machine de dimension n . Chaque processeur possède un numéro " p " dit absolu ($0 \leq p \leq N-1$) qui correspond à une valeur décimale de ce numéro écrit en base binaire (ce qu'on appelle un *code de Gray*) (cf. Figure 4). Les liens entre ces processeurs sont des arrêtes du n -cube, de sorte qu'un processeur ne communique qu'avec ses $n = \log_2(N)$ voisins et, s'il communique avec d'autres non directement voisins, cette communication se fait par des processeurs intermédiaires. On dit que deux processeurs sont physiquement voisins, si et seulement si, en écriture binaire, leurs numéros diffèrent exactement d'un bit (en code de Gray). Un *4-cube* est une machine hypercube comportant 16 processeurs, ce qui correspond à une machine *T-20*. Dans les *T-series* machines, le nombre de processeurs peut varier de $8=2^3$ jusqu'à $16384=2^{14}$ processeurs.

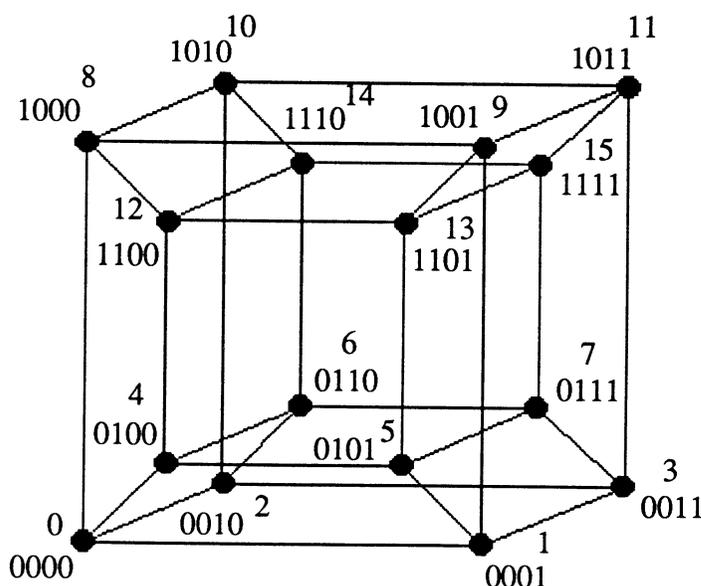


Figure 4 : hypercube T-20 de dimension 4.

La *T-series* machine est une machine, conçue par M. J. Flynn en 1966 (cf. Flynn(66)), *MIMD* (*M*ultiple *I*nstruction *M*ultiple *D*ata) asynchrone, à mémoire distribuée (*un Mégaoctet de mémoire vive pour chaque processeur* (cf.

Tourancheau et al.(88))), i.e. chaque processeur d'une T-series machine a sa propre mémoire accessible que par lui-même, par opposition aux architectures à mémoires partagées où tous les processeurs ont accès à une mémoire centrale.

Comme la mémoire d'une T-series machine n'est pas partagée, les processeurs doivent s'échanger ou se communiquer entre eux. Par suite, chaque processeur envoie un message à un autre processeur à travers au plus $n = \log_2(N)$ liens, N étant le nombre de processeurs de la T-series machine *n-cube*, i.e. une communication se fait par un envoi de messages (cf. *Hwang et al.(84)*).

La T-series de [FPS] est un système de conception modulaire où chaque module est constitué d'un cube de dimension 3 (8 processeurs) relié à un nœud système par un bus (link-bus) et aux autres modules par canaux suivant la topologie hypercube. Le nœud système a pour fonction de gérer, d'une part l'échange entre son module et son disque dur de *Winchester*, d'autre part la communication entre les processeurs de ce module et la machine hôte (ici c'est un Micro-Vax).

Les nœuds système sont reliés entre eux par une structure d'anneau, et chacun d'entre eux possède un disque dur de Winchester. Les entrées/sorties entre le T-40 et le Micro-Vax s'effectuent par un bus connecté au nœud système "0", qui est le processeur privilégié pour les communications vers la machine hôte.

Ce système modulaire permet de construire aisément des systèmes plus importants en augmentant la dimension de l'hypercube (*Le T-200 à Los Alamos, U.S.A*, comprend 128 processeurs).

L'hypercube T-40 est relié à un mini-ordinateur hôte, servant d'interface avec les utilisateurs, qui est un *Micro-Vax* (fabriqué par Digital Equipment Corporation) supportant jusqu'à 7 utilisateurs simultanément.

Le système d'exploitation est *ULTRIX ou VMS* (cf. *UNIX(83)*), et le Micro-Vax est relié au réseau local grenoblois via Ethernet.

• **Les performances théoriques du système :**

- * *au niveau calcul* : la "peak performance" de calcul en parallèle de chaque processeur atteint 12 Mflops (Millions d'opérations flottantes par seconde).
- * *au niveau du transfert de données* : chaque processeur peut recevoir et émettre simultanément 692 Koctets par canal et par seconde (5 Mcoctets pour les 4 canaux en communication bidirectionnelle).

III -3.2 Structure d'un processeur

un processeur, encore appelé noeud de l'hypercube, est composé de 3 parties :

- * **un transputer** (*micro-processeur T-414 fabriqué par INMOS*), spécialisé dans les communications, qui gère les communications entre les processeurs voisins ou avec un nœud système. Il joue aussi le rôle d'unité arithmétique et logique, et assure le contrôle du programme.
- * **une unité de calcul vectoriel VPU** (*Vector Processing Unit*) permet des calculs très rapides en virgules flottantes (grâce à un pipeline sur l'additionneur, le multiplieur et les registres (*cf. Figure 5*)) sur des vecteurs réels codés sur 32 bits et sur 64 bits (format IEEE double précision).
- * **une mémoire vidéo** (*1Moctet*) très performante à double accès aléatoire vers le transputer et série vers les registres du VPU.
- * **un multiplexeur** : pour augmenter la capacité de communication entre les processeurs où chaque processeur a 4 canaux (entrées/sorties) bidirectionnels actifs.

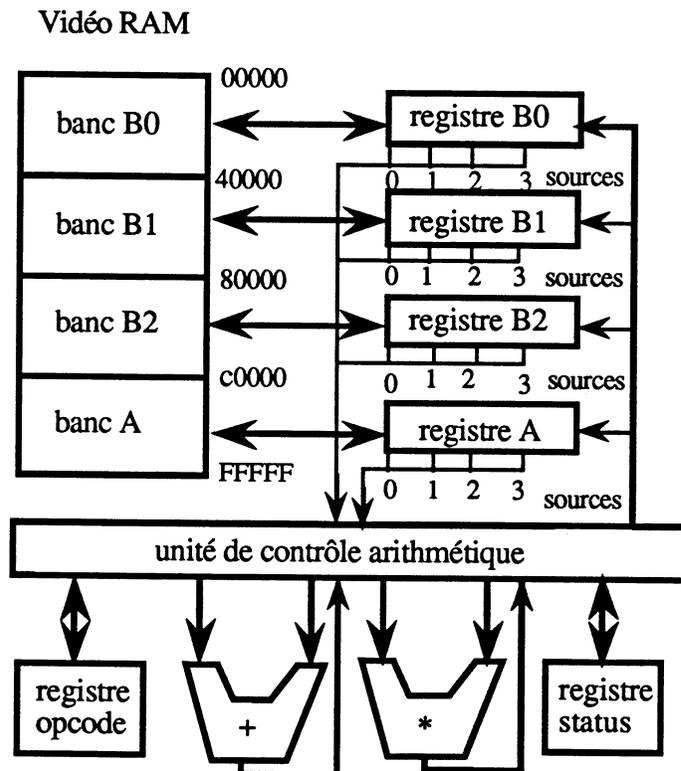


Figure 5 : Organisation des composants du VPU et de la mémoire

Toutes ces composantes sont rangées sur une seule carte, réalisée en circuit VLSI.

III -3.3 Structure d'un nœud système

un nœud système est constitué de 2 parties principales :

- * *une mémoire* (disque dur de Winchester 85 Moctets).
- * *un transputer* (de contrôle).

Le nœud système sert d'interface avec la machine hôte (le Micro-Vax) pour ce qui concerne les entrées/sorties et assure le contrôle des données sur son disque. Lors de l'exécution, le nœud système "0" charge le code reçu du Micro-Vax sur tous les processeurs de son cube et le transmet aux autres nœuds via l'anneau système, chacun le distribuant sur son cube. Par conséquent, tous les processeurs connaissent le même code, mais ils ont cependant un numéro absolu dans l'architecture ; cela leur permet de n'exécuter que la partie du code les concernant grâce à une instruction conditionnelle explicite du programmeur.

III -3.4 Les communications

Sur chaque nœud, la gestion des communications est assurée par le transputer. Celui-ci possède 4 canaux de communications bidirectionnels (émission et réception) (cf. *Figure 6*). Afin d'augmenter la capacité du système, ces 4 sorties ont été multiplexées avec un élément à 4 sorties. Ainsi, chaque processeur dispose donc de 16 canaux de communications ; cependant, seuls 4 d'entre eux peuvent être utilisés simultanément (un multiplexeur ne pouvant avoir qu'une seule de ses entrées actives).

Dans l'hypercube, 2 processeurs sont "*physiquement*" voisins lorsque leur écriture binaire diffère d'un seul bit (*en code de Gray*) (cf. *Y. Saad et al.(88)*).

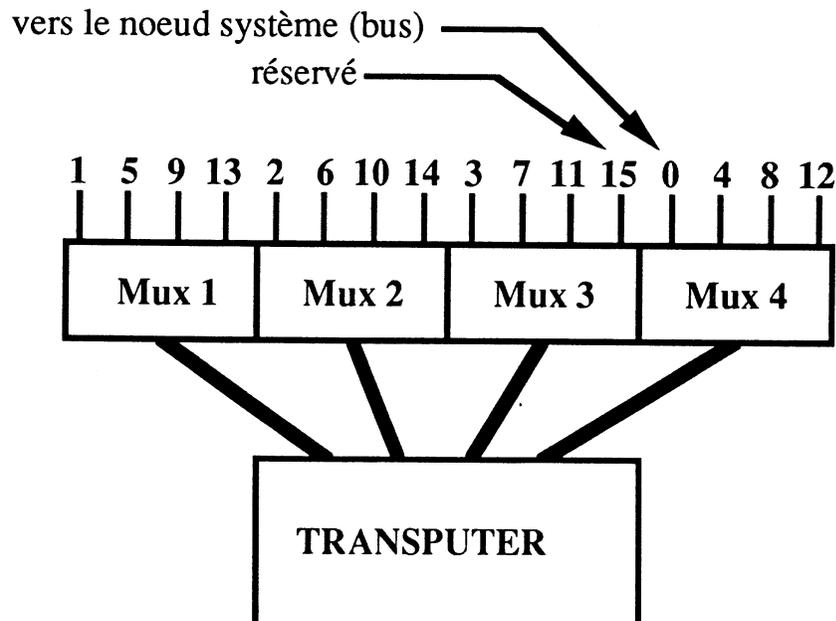


Figure 6 : Transputer et multiplexeurs

Les multiplexeurs étant configurés à chaque communication, on obtient un temps d'initialisation β , qui est beaucoup plus élevé que le temps de transfert des données ; comme le montre le modèle proposé par *Y. Saad 1986*, le coût d'une communication est donné par :

$$t = \beta + \tau n$$

On obtient sur le *T-40* les valeurs $\beta=830$ et $\tau=1.43$ *micro-secondes* (cf. *Kuppuswani et al.(88)*), n étant le nombre d'octets transférés.

* Communication :

Une communication entre processeurs s'effectue en trois étapes :

- i) - Choisir de la configuration,*
- ii) - Définir les canaux de communication,*
- iii) - Utiliser les fonctions de communication.*

i) Topologie :

Les différentes configurations possibles, décrites dans (*Poplawski(88)*), sont :

- Le tore à une dimension (configuration anneau),
- Le tore à deux dimensions (configuration grille),
- L'hypercube de dimension ≤ 5 .

De plus, suivant la topologie, on peut connaître l'emplacement relatif du processeur dans la topologie considérée. Ainsi, dans l'hypercube, les processeurs sont numérotés selon le code de Gray et, dans l'anneau, deux voisins ont des numéros qui diffèrent de 1 bit.

ii) Chemins de communication :

Pour communiquer entre processeurs sur une topologie, il faut définir un chemin qui utilise l'un des 4 canaux physiques. Lors de la définition d'un chemin, on peut spécifier si la communication sera blocante ou non, c'est-à-dire si le processeur doit attendre la fin de la communication avant de continuer à travailler ou s'il peut le faire simultanément.

iii) Fonctions de communication :

Il existe différentes routines de communication permettant chacune d'atteindre un nombre divers de processeurs. On peut ainsi communiquer :

- D'un processeur à un autre (*OTO() : one-to-one*)
- D'un processeur à tous les autres (*OTA() : one-to-all*)
- De tous les processeurs à tous les autres (*ATA() : all-to-all*)

Le transfert des données se faisant octet par octet, il faut donc préciser le nombre d'octets à transférer. Les fonctions de communication ne manipulent que des adresses dans la mémoire.

Si on veut communiquer avec un processeur qui n'est pas voisin immédiat, il faut construire un chemin ou routage, où l'on se déplace d'un bit différent en un bit différent dans l'écriture binaire.

*** Communication Entre Hôte et Processeurs :**

Les communications entre les processeurs d'un cube et le noeud système associé sont assurées par un bus, et sont donc "mono-utilisateur" ; l'accès du noeud système est géré par un logiciel arbitre. Pour effectuer une entrée/sortie, il faut créer la connexion physique entre le processeur et le bus, appeler la routine [*FPS*] d'entrée/sortie qui convient et, enfin, débloquer le multiplexeur. On peut (entre autres) lire ou écrire à l'écran :

- *Des caractères,*
- *des chaînes de caractères,*
- *des entiers,*
- *des réels codés sur 64 bits.*

* Communication entre les nœuds système et les processeurs :

Chacun des nœuds système possède un disque dur de *Winchester* où il est possible pour l'utilisateur de stocker des fichiers (pourvu que ceux-ci soient au format adéquat).

On dispose ainsi des procédures classiques de traitement des fichiers :

- *ouverture du fichier,*
- *lecture dans le fichier,*
- *écriture dans le fichier,*
- *fermeture du fichier.*

III -3.5 La Mémoire

Sur le *T-40*, chaque processeur possède *1 Méga-octet* de mémoire vive d'accès très rapide depuis le *VPU*, ainsi que *2 Kilo-octets* internes au transputer. La mémoire principale est du type *vidéo RAM* à double accès, série depuis le *VPU* et aléatoire depuis le transputer. La mémoire est adressable par mots de *32 bits*, mais les "*blocs*" de transfert peuvent être plus importants (*jusqu'à 1024 octets*).

La mémoire est découpée en quatre bancs égaux, chacun étant associé à un registre vectoriel. L'accès série à la mémoire permet le transfert simultané d'une tranche d'un des bancs mémoire dans les registres vectoriels (*cf. Figure 7*).

Les quatre bancs, numérotés *A, B2, B1, B0*, ont *256 tranches de 1024 octets*, soit *256*128 réels* (un réel étant codé sur *8 octets* en double précision), les adresses allant de *00000 à FFFFF*. Les extrémités de la mémoire (bas du banc *B0* et sommet du banc *A*) sont réservées au système qui les utilise lors des appels des routines "*generic*" ou "*single node*". Le code du programme est chargé dans la partie basse du banc *B0* (au dessus de la tranche réservée au système).

L'accès aléatoire se fait en *3 cycles* du *VPU* pour *4 octets*, soit *13.33 Méga-octets* par seconde (*MO/s*). Il est possible d'effectuer des transferts de blocs ou d'éléments à incrément régulier dans la mémoire.

L'accès série se fait avec les registres vectoriels. A chaque banc de la mémoire est associé un registre. Les bancs sont découpés en *256 tranches de 1024 octets*; lors des accès séries (lecture ou écriture), une tranche complète de *1024 octets* est chargée

(respectivement déchargée) dans un registre (respectivement dans la mémoire). La vitesse de transfert est importante : une tranche de 1024 octets est transmise en 3 cycles du VPU ; on a donc un débit de 3.4 GO/s.

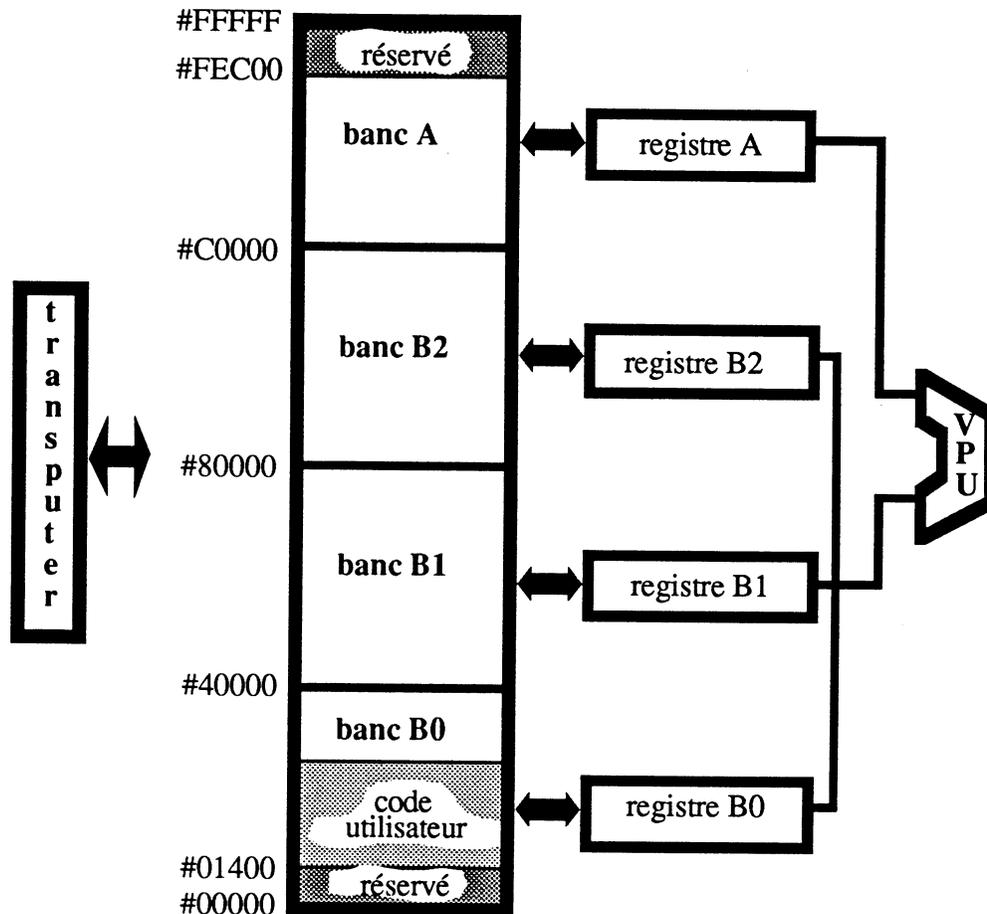


Figure 7 : Schéma de la mémoire et ses accès

III -3.6 L'unité du calcul vectoriel "VPU"

Le VPU (*Vector Processing Unit*) permet des calculs très rapides en virgule flottante (sur des réels codés sur 64 bits, contrairement au transputer qui ne traite ses calculs que sur 32 bits).

Le VPU est conçu pour des opérations vectorielles ou matricielles très rapide. Il a une puissance théorique de l'ordre de 12 MFlops (Millions d'opérations flottantes par seconde) et on peut en pratique arriver à une réalisation de 10 à 11 MFlops.

Le *VPU* contient, d'une part des routines de haut niveau : "*Single Node Routines*" et "*Generic Routines*" qui effectuent des opérations vectorielles complexes telles que l'exponentielle sur des vecteurs (ces routines sont écrites en langage "*Occam*"), d'autre part, des routines de bas niveau, telles que "*Vector Forms*" ou "*Parameter Block Routines*", qui permettent aux utilisateurs de manipuler les composantes du *VPU*. Dans ce niveau, on peut obtenir une meilleure performance, mais ceci nécessite une maîtrise et profonde connaissance du *VPU*. Lors de l'utilisation, toutes les variables en paramètres de ces fonctions doivent être bien placées en mémoire, i.e. dans les fichiers notés : *[Nom_fich].loc*.

III -3.7 Les langages

Les langages disponibles, sont : *Occam*, *Fortran 77* et "*C*" standard auxquels viennent s'ajouter les fonctions *[FPS]* pour les communications et l'accès à l'unité de calcul vectoriel, le "*VPU*" (*Vector Processing Unit*).

III -3.8 Fonctionnement de la *T-series* machine

Une *T-series* machine est une machine mono-utilisateur accessible via un Micro-Vax sous le système d'exploitation *Ulrix* ou *VMS*. Les langages de programmation sont *Occam*, *Fortran 77* et "*C*" standard. Le programme est compilé sous l'allocation du *T-series* par la commande : "*tasg*", pour fournir des codes d'exécution du programme ou encore pour charger le programme dans les processeurs et, ensuite, il faut se désallouer en tapant la commande: "*tdasg*". Chaque processeur identifie son numéro absolu ou son numéro relatif dans la configuration choisie. Les processeurs travaillent d'une manière indépendante ou asynchrone, tant qu'il n'y a pas une communication à faire entre processeurs. Dans les communications, les processeurs se synchronisent et les communications se font par émission de messages à travers des canaux où l'un envoie et l'autre reçoit. Une communication ne peut se réaliser que par un accord mutuel. Chaque processeur possède quatre (4) canaux de communication bidirectionnels (*entrées/sorties*) et chaque canal est multiplexé avec 4 autres canaux. Donc, chaque processeur possède 16 liens possibles de communication, mais au maximum 4 à la fois sont actifs, du fait que le transputer ne supporte que 4 communications parallèles. Comme on l'a vu au paragraphe (III -3.4), les communications coûtent cher ; ainsi, une meilleure réalisation des

communications en temps a été testée par *G. Villard* (cf. *G. Villard (88)*), en montrant qu'en langage *Occam*, la vitesse de communication sur un seul canal peut atteindre 0.937 Moctets/s, lorsque la taille du message est plus d'un Koctets, contre une vitesse normale de l'ordre de 0.66 Moctets/s.

• **Exécution d'un programme :**

Pour lancer un programme, souvent on utilise une exécution en batch, qui consiste à écrire un fichier de commandes, sous l'éditeur *vi*, *ed* ou *emacs*,..., qui contient un ensemble d'instructions, exécutables l'une après l'autre, à effectuer :

Exemple : écriture d'un batch

vi batch.com :

sleep 1800 (exécution des instructions au bout de 1800 secondes *)*

date

tasg (allocation de la T-series *)*

tboot (initialisation de la T-series *)*

MAXTIME 30000 (le temps par défaut est de 3600 secondes *)*

ttrun projet.exe < donnee > sortie.txt

tboot (ré-initialisation de la T-series pour d'autres utilisateurs *)*

tdasg (désallocation de la T-series*)*

date

Pour lancer le batch, on tape : **batch &**

CHAPITRE IV

IMPLANTATION DE RESEAUX NEURONAUX SUR LA MACHINE PARALLELE HYPERCUBE T-40 ET REALISATIONS

IV -1. Introduction

Ce chapitre concerne l'implantation d'un réseau de neurones bidimensionnel sur la machine parallèle hypercube *T-40 [FPS]*. En effet, notre algorithme est prêt à se paralléliser d'une manière simple sur cette machine (cf. *Chap. III et Figure 1*). Donc, on a implémenté (en perfectionnant L. Martel et al. (88)) un programme traitant des réseaux carrés dont les tailles correspondantes sont : 8×8 , 16×16 , 32×32 , 64×64 et 128×128 neurones. On peut aller au delà, mais on risque de dépasser la place mémoire, qui est au total de un Mégaoctet pour chaque processeur. Le programme est écrit en langage "C" standard.

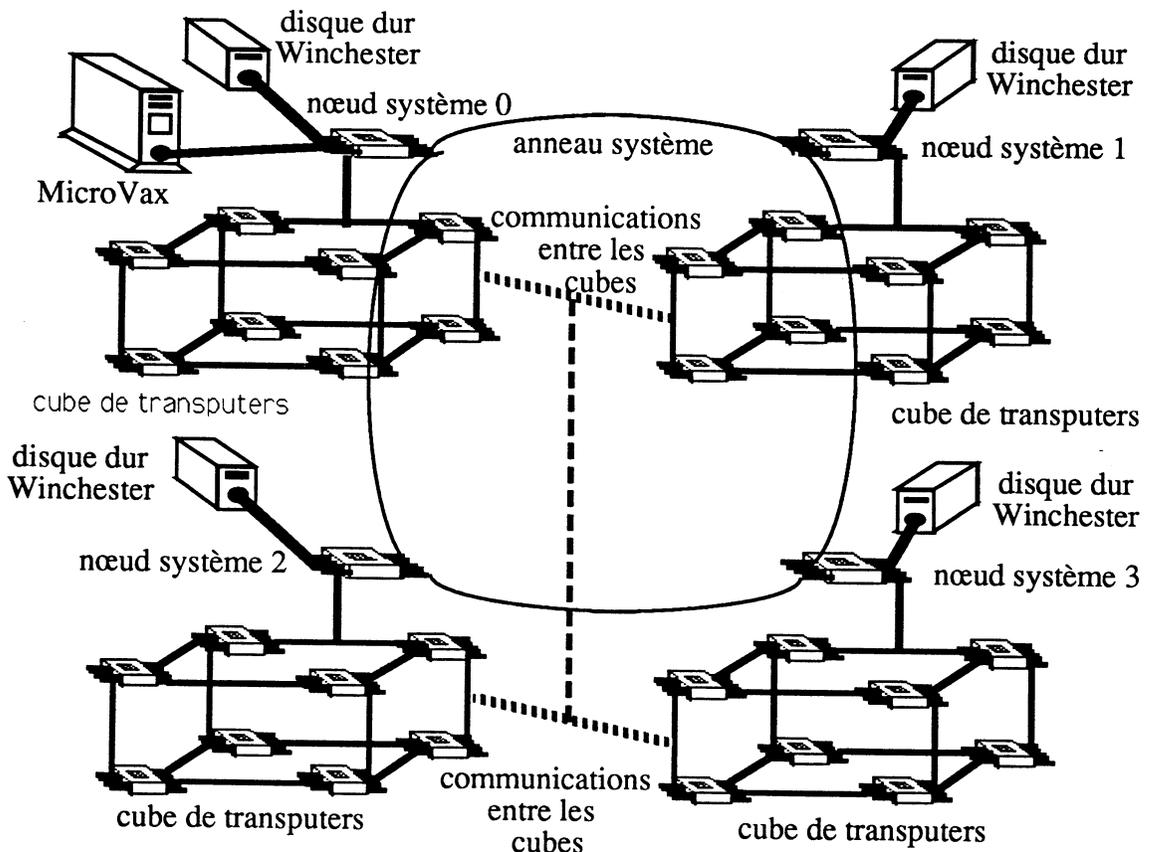


Figure 1 : l'hypercube T-40 [FPS]

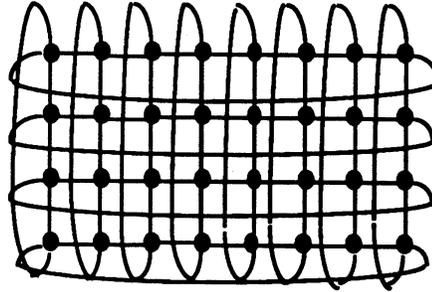


Figure 2 : la grille (4 x 8) à 32 processeurs

IV -2. Implantation de l'algorithme

IV -2.1 Le neurone

D'après ce qui précède, le neurone est connecté à plusieurs neurones voisins (sphères de Manhattan) et, sur ce neurone, il arrive ce qu'on appelle une stimulation extérieure, délivrant une information extérieure dite : "*innovation extérieure*", qui excite le neurone en question (par la fibre d'entrée). Cette innovation est modélisée par un processus de renouvellement ponctuel délivrant un train de spikes selon une loi de densité de probabilité " f_i " qui représente la loi des durées inter-spikes, i.e. la durée entre deux excitations successives, simulée par une réalisation de la loi " f_i " (cf. Figure 9). On a choisi comme type de densité une loi Normale ou une loi rectangulaire dissymétrique (cf. Figure 5, Figure 8). Le neurone ensuite calcule son entrée totale " A_i " (cf. §II -4.2), qui est fonction des états internes " x_j " des neurones voisins, de l'innovation " e_i ", des poids d'interactions " w_{ij} " et du seuil " θ_i ". En sortie, le neurone " i " délivre un spike ou pas, selon la règle d'évolution locale choisie (déterministe ou non-déterministe) (cf. § II -4.2). On définit ainsi la structure de type neurone qui est implémentée par :

```

struct neurone
{
  int   entree ;
  double poids_mahn1[4] ;
  double poids_mahn2[8] ;
  double poids_fibre ;
  double seuil ;
  double refract ;
  double sortie ;
}

```

Par suite, l'ensemble de neurones interconnectés forme ce qu'on appelle un réseau de neurones, qui est décrit dans le paragraphe suivant.

IV -2.2 Le réseau de neurones

Un réseau est donc un ensemble de neurones interconnectés. Soit $\Omega=[1..taille] \times [1..taille]$ la sous-boîte carrée sur lequel notre réseau est implémenté, comprenant $N=(taille)^2$ neurones. On a implémenté le réseau sur trois types de configurations de l'hypercube [FPS] T-40, cela pour une raison bien objective : l'étude du degré de parallélisme de l'algorithme du réseau. On a trouvé que, pour les réseaux de taille supérieure ou égale à $32 \times 32 = 1024$ neurones, le speed-up (ou l'accélération) par rapport au temps d'exécution sur un seul processeur (i.e. temps séquentiel) dépend linéairement du nombre de processeurs (cf. §IV -3.1). Les configurations sont des tores à deux dimensions comportant 4 (resp. 16 et 32) processeurs, de sorte qu'on confie à chaque processeur $N/4$ (resp. $N/16$ et $N/32$) neurones, autrement dit que chaque processeur gère une sous-boîte $[taille/2] \times [taille/2]$ (resp. $[taille/4] \times [taille/4]$ et $[taille/4] \times [taille/8]$).

Chaque processeur dispose d'une taille maximale fixée à 32×32 , à cause de la place mémoire du processeur qui est au total de 1 Mégaoctet (cf. chap. III). On en déduit la structure du réseau implémentée par : `struct neurone reseau[32][32]` ;

Ensuite, nous allons décrire chacun des éléments du réseau en détail :

IV -2.3 Génération des entrées

IV -2.3.1 Génération d'un train de spikes

* Rappel théorique :

Soit "Y" une variable aléatoire de loi uniforme sur $[0, 1]$ et "y" une valeur observée par cette variable (cf. Figure 3).

Posons $X = F^{-1}(Y)$, où "F" est la fonction de répartition supposée bijective de la densité de probabilité "f", alors la loi de la variable aléatoire "X" suit bien la densité donnée par "f".

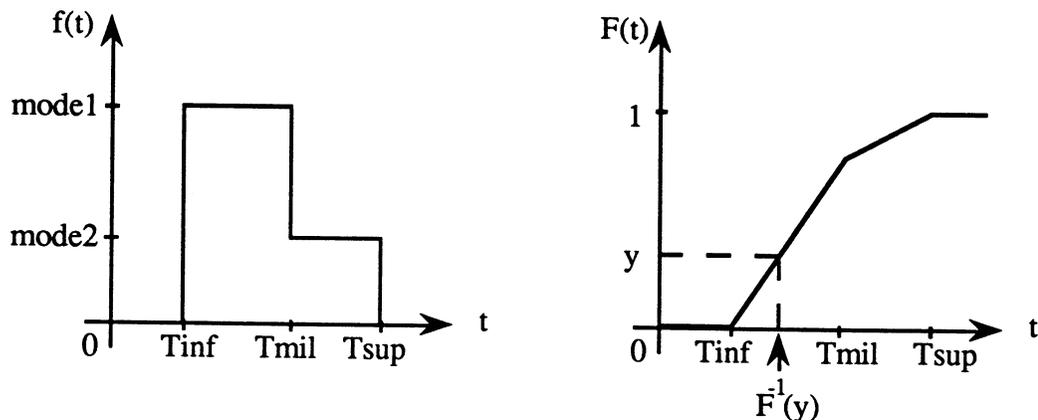


Figure 3 : La densité dissymétrique unimodale "f" et sa fonction de répartition "F"

En effet, calculons la fonction de répartition de " X ", notée " $F_X(t)$ " :

$$\begin{aligned}
 F_X(t) &= \text{Prob}\{ X \leq t \} \\
 \Rightarrow F_X(t) &= \text{Prob}\{ F^{-1}(Y) \leq t \} \\
 \Rightarrow F_X(t) &= \text{Prob}\{ Y \leq F(t) \} \quad (\text{car } F \text{ est croissante}) \\
 \Rightarrow F_X(t) &= F(t) ; \forall t \in \mathbb{R} \quad (\text{car } Y \text{ suit une loi uniforme sur } [0, 1])
 \end{aligned}$$

Par conséquent : $F_X = F$ et donc $f_X = f$, ce qui implique que la loi de " X " suit bien la loi de densité de probabilité " f ".

*** La technique de génération :**

On génère ainsi un train de spikes à l'aide d'un processus de renouvellement ponctuel, tel que sa densité de renouvellement est " f " (cf. Figure 4), qui représente la loi des intervalles inter-spikes. Or, la réalisation du premier intervalle inter-spike joue un rôle important dans la stationnarité du processus de renouvellement en question.

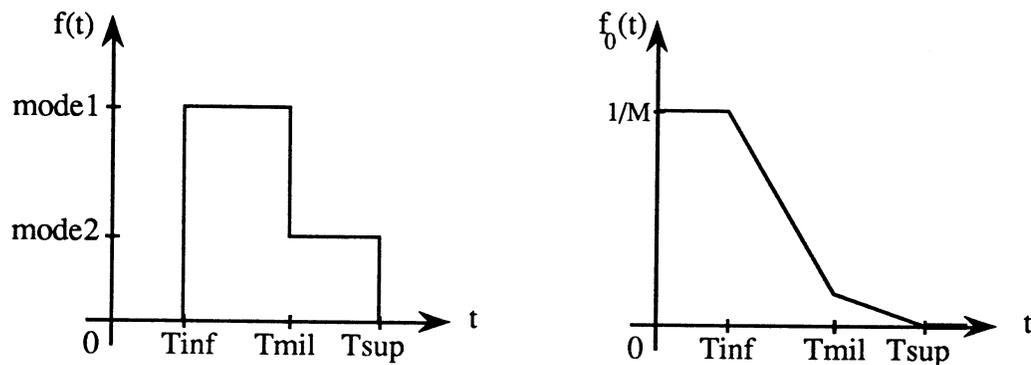


Figure 4 : densités de probabilités " f " et " f_0 "

Pour cela, la génération du premier intervalle inter-spike doit être traitée séparément. On choisit donc " $f_0(t)$ " la densité de probabilité générant ce premier intervalle, telle que (cf. Figure 4) :

$$f_0(t) = \frac{1 - F(t)}{M}, \quad \text{où } M = \int_{-\infty}^{+\infty} t f(t) dt = \int_0^{+\infty} [1 - F(t)] dt$$

On a ainsi implémenté deux densités différentes, la première est la densité Normale (ou de Gauss), notée : $N(M, \sigma)$, et l'autre est la densité rectangulaire dissymétrique, notée : $d(m)$.

Soit " $F(t)$ " (resp. " $F_0(t)$ ") une fonction de répartition de la densité de renouvellement " $f(t)$ " (resp. " $f_0(t)$ ") ; étant donné { " y " } une série de réalisations de la variable aléatoire " Y " de loi uniforme sur [0, 1], alors, à chaque " y ", on associe un

intervalle de durée "x" telle que $x=F^{-1}(y)$. Dans la pratique, "y" est généré par un générateur pseudo-aléatoire sur $[0, 1]$, celui du "VPU", par appel de la routine suivante :

" Single Node Routines " : VRAND (&t , &y, 1, 1)
(Vector Random Number Generator) [T-S].

Tout le problème réside dans l'inversion de la fonction de répartition "F" (resp. "F₀"). Néanmoins, dans la pratique, on applique une méthode dite : "méthode dichotomique", pour passer de "y" à "x", qui sera détaillée par la suite. Cependant, la qualité de cette technique de génération repose essentiellement sur la finesse de l'échantillonnage de la fonction de densité "f(t)" (resp. "f₀(t)") et surtout de la qualité du générateur des nombres pseudo-aléatoires.

- **La loi normale** : $N (M , \sigma)$
 de moyenne " M " et d'écart-type " σ " (cf. Figure 5)

$$f (t) = \frac{1}{\sqrt{2\pi} \sigma} \exp \left[- \frac{(t - M)^2}{2 \sigma^2} \right]$$

Remarque : On choisit " $M > 3 \sigma$ " afin de correspondre à une réalité neuro-biologique. On prend le temps positif, puisque la partie négative de la loi normale n'a aucune signification physique.

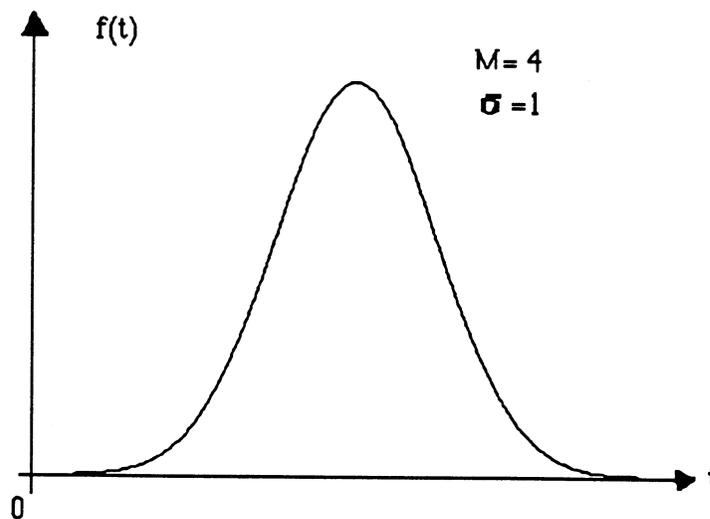


Figure 5 : La loi de Gauss

Environ 99.7% de l'information de la loi gaussienne est contenue dans l'intervalle $[M-3\sigma, M+3\sigma]$; comme on a supposé que $M > 3\sigma$ (cf. remarque ci-dessus), nous choisissons l'intervalle des réalisations comme suit : $[0, M+3\sigma]$.

• **Calcul des inverses de F et de F_0 :**

1) Il se fait par saisie graphique (dite : tablette digitale) de la courbe de la densité de Gauss " f ", i.e. on prend un pas " δ " d'échantillonnage de " f " de l'ordre de : $(M+3\sigma)$ divisé par 1000. Ensuite, on calcule les valeurs de la fonction de répartition " F " correspondant aux abscisses discrètes $0, \delta, 2\delta, \dots, 1000\delta$, qui est résumé par l'algorithme suivant (cf. Figure 6):

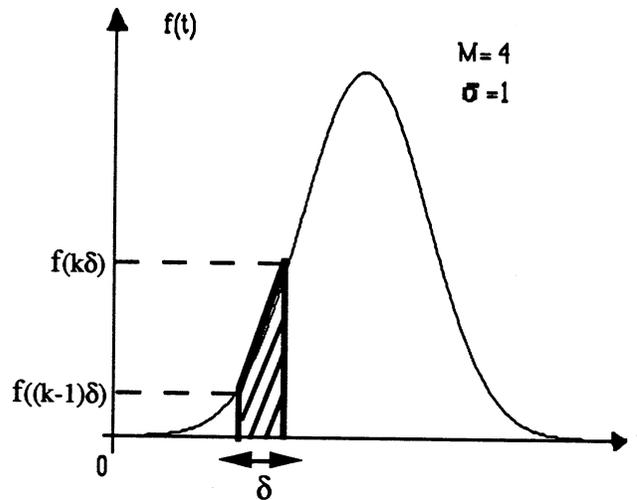


Figure 6 : Echantillonnage de la densité " f "

• **Fonct_repart ()**

Début

$$f(0) = 0 ;$$

$$I(0) = 0 ;$$

$$F(0) = 0 ;$$

Boucle : For ($k=1$; $k \leq 1000$; $k++$)

Début

$$F(k\delta) = I(k-1) + \delta[f((k-1)\delta) + f(k\delta)]/2;$$

$$I(k) = F(k\delta) ;$$

Fin

Fin

En ce qui concerne les valeurs de " F_0 ", elles sont données par la relation suivante:

$$F_0(k\delta) = [1 - F(k\delta)] / M, \text{ pour tout } k \in \{0, \dots, 1000\}.$$

Enfin, on rassemble ces trois informations dans un tableau: " $tab_fonction[1001]$ " qui contient l'abscisse, l'ordonnée de la fonction de répartition " F " et l'ordonnée de la fonction de répartition " F_0 ". d'où l'implémentation (cf. Figure 7) :

```

struct image
{
  double abscisse ;
  double ordonnee ;
  double ordonnee0 ;
} tab_fonction[1001] ;

```

x1	F(x1)	F0(x1)
x1000	F(x1000)	F0(x1000)

Figure 7 : *tab_fonction*[1001]

2) Ce calcul des inverses de "F" ou de "F₀" se fait par une recherche dite : *dichotomique* dans la table de discrétisation donnée par " *tab_fonction*[1001]". On décrit cette méthode par l'algorithme récursif ci-après :

• *Fonct_1*(min, max, y)

Début

$p = 1000/2$;

Si $F(x_p) \geq y$ **alors**

Début

Si $F(x_{p-1}) \leq y$ **alors**

Début

Si y est proche de $F(x_{p-1})$ **alors** {résultat = x_{p-1} }

Sinon {résultat = x_p }

Fin

Sinon {résultat = *Fonct_1*(min, p, y)}

Fin

Sinon

Début

Si $F(x_{p+1}) \geq y$ **alors**

Début

Si y est proche de $F(x_{p+1})$ **alors** {résultat = x_{p+1} }

Sinon {résultat = x_p }

Fin

Sinon {résultat = *Fonct_1*(p, max, y)}

Fin

Fonct_1(min, max, y) = résultat ;

Fin

(* Le procédé dichotomique *)

• Fonct_repart_1(y)

Début

Si $y \geq F(x_{1000})$ alors $F_{-1}(y) = x_{1000}$;

Sinon $F_{-1}(y) = \text{Fonct_1}(0, 1000, y)$;

Fin

De même pour " F_0^{-1} ", il suffit de remplacer " F " par " F_0 " dans l'algorithme précédent.

• La loi rectangulaire dissymétrique (bimodale)

définie par (cf. Figure 8) :

$$f(t) = \begin{cases} 0, & \text{si } t < T_{\text{inf}} \\ \text{mode1}, & \text{si } T_{\text{inf}} \leq t < T_{\text{mil}} \\ \text{mode2}, & \text{si } T_{\text{mil}} \leq t < T_{\text{sup}} \\ 0, & \text{si } t \geq T_{\text{sup}} \end{cases}$$

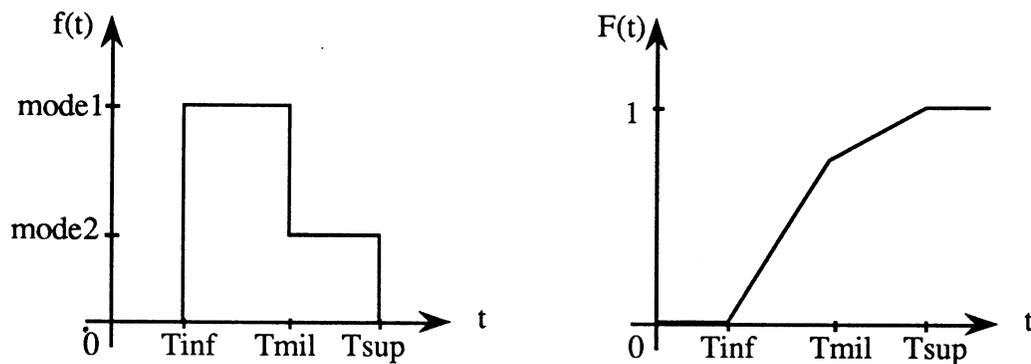


Figure 8 : loi rectangulaire dissymétrique

"f" est telle que : $\text{mode1} \cdot (T_{\text{mil}} - T_{\text{inf}}) + \text{mode2} \cdot (T_{\text{sup}} - T_{\text{mil}}) = 1$ (*)

cela parce que "f(t)" est une densité de probabilité, c'est-à-dire :

$$\int_0^{+\infty} f(t) dt = 1 \text{ et sa moyenne est donnée par : } m = \int_{-\infty}^{+\infty} t f(t) dt = \int_0^{+\infty} [1 - F(t)] dt$$

Nous conservons les paramètres : *mode1*, *Tinf*, *Tmil* et *Tsup*, puisque le cinquième paramètre *mode2* est déduit directement par la relation (*).

• **Calcul des inverses de F et de F_0 :**

1) On procède de la même façon que celle décrite dans le cas de la densité de Gauss, sauf en ce qui concerne le pas d'échantillonnage δ , qui est égal à $T_{sup} / 1000$. Les valeurs de la fonction de répartition " F ", correspondant aux abscisses discrètes T_{inf} , $T_{inf} + \delta$, $T_{inf} + 2\delta$, ..., $T_{inf} + 1000\delta$, sont données d'une manière précise, car on connaît explicitement la fonction " F ", qui définie par (cf. Figure 8) :

$$F(t) = \begin{cases} 0, & \text{si } t < T_{inf} \\ mode1.(t - T_{inf}), & \text{si } T_{inf} \leq t < T_{mil} \\ mode2(T_{mil} - T_{inf}) + mode2(t - T_{mil}), & \text{si } T_{mil} \leq t < T_{sup} \\ 0, & \text{si } t \geq T_{sup} \end{cases}$$

En ce qui concerne les valeurs de " F_0 ", elles sont données par la relation suivante :

$$F_0(k\delta) = [1 - F(k\delta)] / m, \text{ pour tout } k \in \{0, \dots, 1000\}.$$

On aura par suite un tableau " $tab_fonction[1001]$ " contenant abscisse, ordonnée de " F " et ordonnée de " F_0 ".

2) Ce calcul des inverses de " F " ou de " F_0 " se fait de la même façon que précédemment (cf. cas de la loi gaussienne), c'est-à-dire par une recherche dite dichotomique, dans la table de discrétisation donnée par " $tab_fonction[1001]$ ". Néanmoins, on peut implémenter directement " F^{-1} " par son expression explicite, qui est la suivante :

$$F^{-1}(t) = \begin{cases} T_{inf}, & \text{si } t = 0 \\ T_{inf} + \frac{t}{mode1}, & \text{si } 0 < t < mode1(T_{mil} - T_{inf}) \\ T_{mil} + \left[\frac{t - mode1(T_{mil} - T_{inf})}{mode2} \right], & \text{si } mode1(T_{mil} - T_{inf}) \leq t < 1 \\ T_{sup}, & \text{si } t = 1 \end{cases}$$

• **Calcul des entrées $\{ e_i(t) \}$:** (pour " i " fixé)

On obtient le train de spikes suivant (cf. Figure 9) :

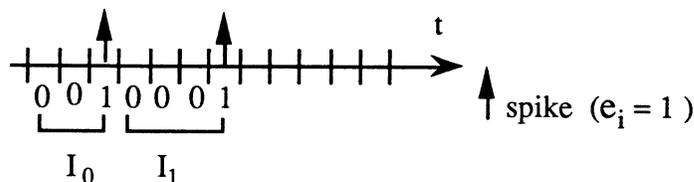


Figure 9 : Train de spikes

- Initialisation du processus générant les entrées :

En effet, soit "y" un nombre tiré au hasard selon la loi uniforme sur $[0, 1]$, donnée par la routine "VRAND (&t, &y, 1, 1)", et soit I_0 (cf. Figure 9) le premier intervalle inter-spike généré par la loi "f₀", i.e. I_0 est l'entier le plus proche du réel " $F_0^{-1}(y)/\Delta t$ ", où Δt est le pas d'échantillonnage du réseau. Pour transformer un réel en entier le plus proche de lui, on a écrit une fonction "arrondi()" en utilisant une routine dans le "VPU" : "GENERIC ROUTINES" : VO(&loc_reel, AF_XFIX, loc_ent, VP_NULL, 1)
(Vector Operation) [T-S].

• **arrondi()**

Début

```
reel_inter_spike = F0-1(y)/Δt ;
VO(&reel_inter_spike, AF_XFIX, entier, VP_NULL, 1) ;
test = reel_inter_spike - entier ;
Si test ≤ 0.5 alors {résultat = entier} ;
Sinon {résultat = entier + 1} ;
```

Fin

• **init_inter_spike()**

Début

```
VRAND(&t, &y, 1, 1) ;
reel_inter_spike = F0-1(y)/Δt ;
comptr_intervalle = arrondi( ) ;
I0 = comptr_intervalle ;
ei(0) = 0 ; (* l'entrée du neurone "i" à l'instant '0' *)
```

Fin

- L'algorithme générant les entrées :

• **gener_train_spikes()**

Début

```
init_inter_spike( ) ;
Boucle ( 1 ≤ t ≤ nb_iter_max )
```

Début

```
Si comptr_intervalle = 0 alors
```

Début

```
(* on tire un autre nombre "y" pseudo-aléatoire *)
VRAND(&t, &y, 1, 1) ;
reel_inter_spike = F-1(y)/Δt ;
comptr_intervalle = arrondi( ) ;
It = comptr_intervalle ;
```

```
ei(t) = 1 ; (* l'entrée du neurone "i" à l'instant 't' *)
```

Fin

Sinon

Début

```
comptr_intervalle = comptr_intervalle - 1 ;
```

```
ei(t) = 0 ;
```

Fin

Fin

Fin

IV -2.4 Connexions et Communications

Notre réseau est une grande matrice subdivisée en 4 (*resp. 16 et 32*) sous-boîtes, chacune correspondant à un processeur. L'idée d'utiliser une grille à deux dimensions projetée sur les 4 (*resp. 16 et 32*) processeurs de l'hypercube est inspirée par la structure plane du réseau traité. Comme le réseau est un ensemble de neurones interconnectés, ce découpage en sous-boîtes nécessite donc des communications entre les processeurs. En effet, par découpe et répartition du réseau carré initial en sous-boîtes plus petites sur les processeurs et du fait qu'un neurone évolue en fonction des états internes de ses neurones voisins (sphères de Manhattan), certains neurones ont des voisins dans les processeurs directement voisins (grille à deux dimensions donc quatre voisins directs), mais aussi dans un processeur non directement voisin distant de deux étapes de communications. Ce type de communication n'est nécessaire que pour les quatre coins de la sous-boîte dévolue à chaque processeur.

Toutefois, il faut limiter ces communications qui coûtent cher. Le temps de communication est " $\beta + \tau n$ micro-secondes", " β " étant le temps d'initialisation et " n " étant le nombre d'octets, où $\beta=830$ micro-secondes et $\tau=1.43$ micro-secondes. Etant donné un neurone " i ", l'information qu'on doit récupérer est l'état interne des neurones voisins $N(i)$ (sphères de Manhattan) à la date ' k ' pour que le neurone puisse calculer son état de sortie à la date ' $k+1$ '. On comprend donc qu'à l'itération ' k ', on doit stocker les états internes des neurones appartenant au voisinage considéré, pour préparer l'itération ' $k+1$ '.

On constate que beaucoup de neurones possèdent des voisins en commun à l'extérieur comme à l'intérieur du processeur. De plus, le coût de communication étant cher en temps, on a décidé de communiquer l'information sous forme compacte au lieu d'effectuer une distinction pour chaque neurone ou encore en les stockant dans un tableau noté :

int memor[36][36]

dont la taille inclut non seulement la sous-matrice *reseau[32][32]* traitée par chaque processeur, mais encore :

- les quatre bandes périphériques de largeur deux où seront recopiées les valeurs de sorties des neurones des processeurs voisins, suite à une communication d'ordre un.
- les quatre valeurs de sortie des voisins de la sphère de rayon 2 des quatre neurones de coin du processeur, suite à une communication d'ordre deux (en deux étapes).

Par conséquent, on aura, à chaque itération, une fois les communication entre les processeurs effectuées, toute l'information nécessaire au traitement des neurones d'un processeur stockée dans le tableau "memor[36][36]".

Pour faciliter le mode de programmation, on a cherché un codage astucieux pour retrouver les coordonnées du $k^{\text{ème}}$ neurone voisin (i, j) à partir des coordonnées (x, y) du neurone à étudier (cf. figure 10). En effet, soit un neurone de coordonnées (x, y) dans le sous-réseau "reseau[32][32]", et soit (i, j) les coordonnées du neurone voisin "numéro k " sur la sphère de rayon 1 ou 2. Pour retrouver les nouvelles coordonnées dans la matrice "memor[36][36]", il suffit d'ajouter 2 unités à chacune des coordonnées du sous-réseau "reseau[32][32]".

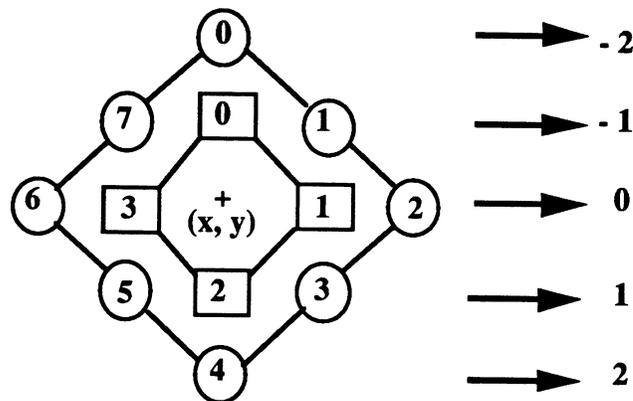


Figure 10 : code des coordonnées du neurone $N^{\circ} k$

• formulation en langage "C" :

```
static init_code (
{
  int k ;
  code_mahn1[0] = -1 ;
  code_mahn1[1] = 0 ;
  code_mahn1[2] = 1 ;
  code_mahn1[3] = 0 ;
  code_mahn2[0] = -2 ;
  for ( k = 1; k <= 4; k++ ) code_mahn2[k] = code_mahn2[k-1] + 1 ;
  for ( k = 5; k <= 7; k++ ) code_mahn2[k] = code_mahn2[k-1] - 1 ;
}
```

Les coordonnées (i, j) du neurone voisin $N^{\circ} k$ de la sphère de rayon 1 dans la table memor[36][36] sont :

$$\begin{cases} i = x + 2 + \text{code_mahn1}[k] \\ j = y + 2 + \text{code_mahn1}[(k+1)\text{mod}4] \end{cases}$$

Les coordonnées (i, j) du neurone voisin N° k de la sphère de rayon 2 dans la table *memor[36][36]* sont :

$$\begin{cases} i = x + 2 + \text{code_mahn2}[k] \\ j = y + 2 + \text{code_mahn2}[(k+2)\text{mod}8] \end{cases}$$

*** Fonctions de communication :**

On a utilisé deux types de fonctions de communication :

- i)- Les communications ONE-TO-ONE : " *OTO_X ()* "
- ii)- Les communications ONE-TO-ALL : " *OTA_ID ()* "

i) *Les communications ONE-TO-ONE* : " *OTO_X ()* "

Il s'agit de communications entre deux processeurs. Une communication ne peut s'effectuer que sous les conditions suivantes (cf. chap.III) :

- Définir une topologie sous laquelle l'hypercube est configuré,
- Définir les canaux de la communication,
- Utiliser les fonctions de communication.

• Définir la topologie que l'on va utiliser :

- Tore à une dimension (configuration anneau), défini par :
config_torus_1d (Taille, &descript)
- Tore à deux dimension (configuration grille), défini par :
config_torus_2d (Taille_1,Taille_2, &descript)
- Hypercube, défini par :
config_hyp (dimension, &descript)

où

Taille : le nombre de processeurs (en puissance de 2) dans la configuration.

Taille_1 : le nombre de processeurs (en puissance de 2) dans une dimension de la configuration.

Taille_2 : le nombre de processeurs (en puissance de 2) dans une autre dimension de la configuration.

descript : le descripteur de la topologie.

& : signifie l'adresse d'une variable en langage 'C'.

• Définir les canaux de la communication :

lien = open_l (descript, param, mode)

• Appeler les fonctions de communication :

OTO_X (lien, dim, depar, arri, &depar, &arri, N),

OTA_1D(lien, dim, depar, &depar, &arri, N).

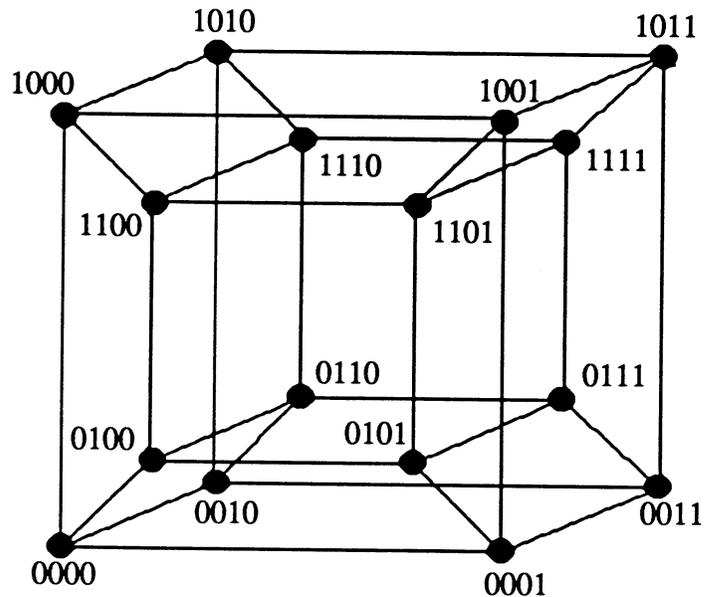


Figure 11 : codes de Gray d'un hypercube T - 20

Dans le cas où on configure l'hypercube en grille "4 x 4", on construit nous-même les emplacements absolus des processeurs, en respectant la condition qu'on va rappeler (cf. Figure 11) :

"Deux processeurs sont voisins 'physiquement', lorsque leurs numéros en écriture binaire ne diffèrent que d'un seul bit (règle connue sous le nom code de gray)".

La formulation en langage 'C' :

• cas de grille (2 x 2) :

```

init_config_tore_1 ( )
{
  char *td_hyp ;
  int line ;
  config_hyp( 2, &td_hyp ) ;
  line = open_l ( td_hyp, 2, OTOX ) ;
}

```

• cas de grille (4 x 4) :

```

init_config_tore ( )
{
  char *td_hyp ;
  int line ;
  config_hyp(4, &td_hyp) ;
  init_grille ( ) ;
  line = open_l(td_hyp, 2, OTOX) ;
}

```

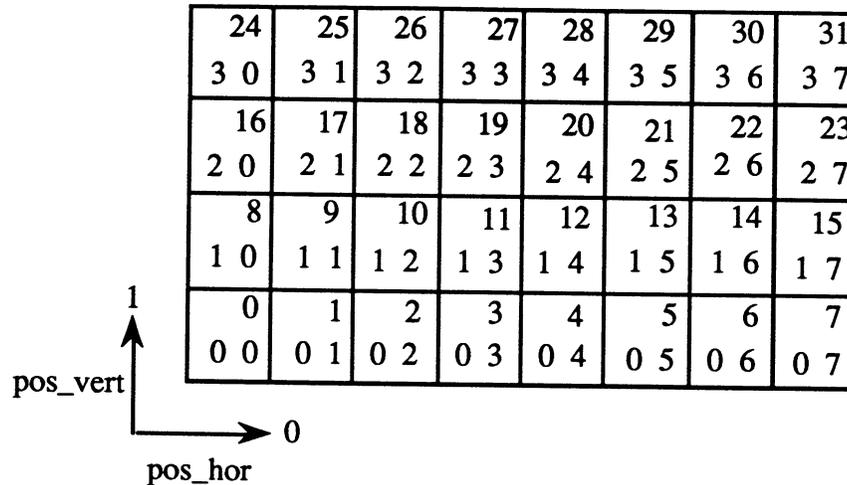


Figure 12 : positions et coordonnées des processeurs dans la grille (4 x 8)

• cas de grille (4 x 8) : (cf. Figure 12)

```

init_config_tore_3 ( )
{
  char *td_tore ;
  int line ;
  int pos_vert, pos_hor, pos_proc ;
  config_torus_2d( 8, 4, &td_tore ) ;
  /* pour la gestion des communications sur le tore_2d */
  line = open_l(td_tore, 0, OTOX) ;
  /* position relative du processeur dans le tore_2d */
  pos_vert = config_pos(td_tore, 1) ;
  pos_hor = config_pos(td_tore, 0) ;
  /* numérotage (relatif) des processeurs par ligne */
  pos_proc = pos_vert * 8 + pos_hor ;
}

```

Note : Pour le premier [resp. le deuxième] cas, nous voulons configurer une grille de 4 [resp. 16] processeurs, or l'hypercube "T-40" configure automatiquement quatre [resp. deux] grilles à 4 [resp. 16] processeurs. Mais, grâce à une instruction conditionnelle dans le programme, on utilisera une seule grille.

0 0000	4 0100	12 1100	8 1000
1 0001	5 0101	13 1101	9 1001
3 0011	7 0111	15 1111	11 1011
2 0010	6 0110	14 1110	10 1010

Figure 13 : grille (4 x 4) par codes de Gray

En ce qui concerne le cas de la grille (4 x 4) (cf. Figure 13), chaque processeur, sous la topologie hypercube, possède quatre directions de communication (4 dimensions), allant de 0 à 3 ; pour définir la valeur d'une direction d'un processeur par rapport aux voisins, il suffit de lui associer le numéro du bit qui diffère entre chacun d'eux, soit alors : *dim_dessus*, *dim_dessous*, *dim_droite* et *dim_gauche*. Par exemple, en définissant les directions du processeur N° 14 dans la grille ci-dessus, il vient :

dim_dessus = 0
dim_dessous = 1
dim_droite = 2
dim_gauche = 3

Il reste ensuite à définir la position relative d'un processeur sur l'une de ses dimensions de communication : c'est la valeur du bit correspondant à cette dimension. Soit alors : *pos_dessus*, *pos_dessous*, *pos_droite* et *pos_gauche* ; ce sont des variables définissant les quatre positions relatives d'un processeur sur les dimensions définies ci-dessus. Il vient par exemple, toujours le processeur N° 14 :

pos_dessus = 0
pos_dessous = 1
pos_droite = 1
pos_gauche = 1

0 0	0 1	0 2	0 3
1 0	1 1	1 2	1 3
2 0	2 1	2 2	2 3
3 0	3 1	3 2	3 3

Figure 14 : coordonnées des processeurs (*pos_vert*, *pos_hor*) dans la grille (4x4) à 16 processeurs

Finalement, on a associé à chaque processeur des coordonnées `pos_vert` et `pos_hor` afin d'identifier ce processeur dans la grille considérée (cf. *Figure 14*), telles que la première composante signifie sa position verticale et l'autre sa composante horizontale. Par exemple, le processeur N° 14 a pour coordonnées :

```
pos_vert = 3
pos_hor = 2
```

Remarque : En ce qui concerne la construction de la grille (4x8), on a utilisé la grille propre à l'hypercube T-40 "config_torus_2d()", tout en manipulant seulement les positions relatives des processeurs dans le tore (cf. § IV - 2.4).

Après avoir construit cette topologie de la grille en question, on est capable donc d'appeler la fonction de communication citée plus haut `OTO_X ()`, en précisant ses paramètres comme suit :

- Le lien de communication (les canaux de communication),
- La direction de communication (représentée par : `dim_dessus, ...`),
- L'emplacement (relatif) du processeur émetteur sur cette direction ,
- L'emplacement (relatif) du processeur récepteur sur cette direction ,
- L'adresse de départ de la donnée à transférer ,
- L'adresse d'arrivée de la donnée à recevoir ,
- Le nombre d'octets à transférer .

Une telle communication est valable si et seulement si les processeurs en question sont mutuellement informés et également d'accord, c'est-à-dire si l'un précise la quantité d'envoi et l'autre attend cette quantité. Illustrons cela par l'exemple suivant : une communication entre le processeur N° 14 et le processeur N° 10 ; supposons que le processeur N° 14 envoie ses quatre dernières valeurs du tableau "`int T[10]`" à valeurs entières au processeur N° 10 dans le même tableau "`T[10]`", on a alors :

```
/* procédure de communication à droite */
proc_droite ( )
{
  IF ( posabs = 14 )
  {
    /* Envoie du processeur N°14 */
    OTO_X(line, dim_droite, pos_droite, 1-pos_droite, &T[6], &T[0],4x4);
    wait_l(line) ;
  }
  ELSE {
    /* Réception du processeur N°10 */
    OTO_X(line, dim_gauche, 1-pos_gauche,pos_gauche, &T[6], &T[0],4x4);
    /* ne pas faire d'autres communications */
    /* tant que ce n'est pas terminée */
    wait_l(line) ;
  }
}
```

ii) Les communications ONE-TO-ALL : "OTA_ID "

La topologie utilisée ici est la procédure [cf. (i)] :

`config_torus_1d (Nombre , &td_anneau)`

Nombre = 16 ou 32 selon le cas traité (cf. Figure 15).

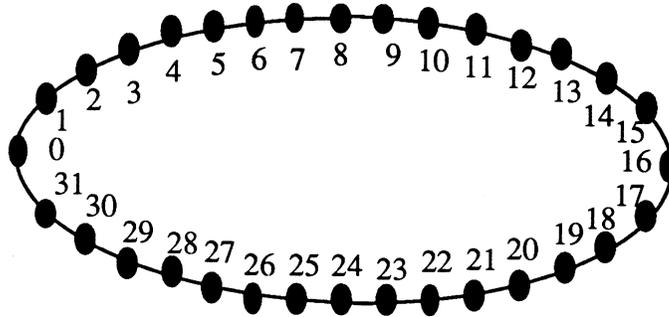


Figure 15 : configuration d'anneau à 32 processeurs

On a confié au processeur N° 0 le soin de transmettre l'ensemble des paramètres communs à tous les processeurs tels que : la taille du réseau, le critère d'évolution du réseau, ..., puisque c'est le processeur commun dans toutes les configurations choisies par l'utilisateur.

Du fait du même principe que dans i), pour utiliser la fonction de communication "OTA_ID", il faut choisir la configuration (cf. Figure 15) et puis définir les canaux de communication. Il s'agit en fait de la communication d'un processeur à tous les autres processeurs, ce qui implique l'écriture suivante :

La formulation en langage "C" :

• cas de grille (2 x 2) :

```
init_config_anneau_1( )
{
  char *td_anneau ;
  int line_anneau , pos ;
  config_torus_1d (4, &td_anneau) ;
  pos = config_pos ( td_anneau , 0) ;
  line_anneau = open_l(td_anneau, 2, 0X0002);
}
```

• cas de grille (4 x 4) :

```

init_config_anneau_2 ( )
{
  char *td_anneau ;
  int line_anneau , pos ;
  config_torus_1d (16, &td_anneau) ;
  pos = config_pos ( td_anneau , 0) ;
  line_anneau = open_l(td_anneau, 2, 0X0002);
}

```

• cas de grille (4 x 8) :

```

init_config_anneau_3 ( )
{
  char *td_anneau ;
  int line_anneau , pos ;
  config_torus_1d (32, &td_anneau) ;
  pos = config_pos ( td_anneau , 0) ;
  line_anneau = open_l(td_anneau, 2, 0X0002);
}

```

Remarque : On peut ne pas utiliser ces procédures "configurations d'anneaux" décrites ci-dessus, cela nous permet de gagner du temps au niveau des communications entre processeurs, en faisant lire les données par tous les processeurs.

Dans le cas où le processeur "0" lit les données, puis les transmet aux autres processeurs selon la configuration d'anneau donnée, on peut l'expliquer comme suit. Si le processeur envoie par exemple la valeur "taille" (la taille du réseau traité) à tous les processeurs dans l'anneau comprenant 16 processeurs. On fait donc appel à la fonction suivante :

main ()

```

{
  (*partie déclaration *)
  init_config_anneau ( )
  IF ( pos = 0 )
  {
    /* Envoie du processeur N° 0 */
    OTA_ID(line_anneau, 0, 0, &taille, &taille, 4 ) ;
    wait_l ( line_anneau ) ;
  }
  ELSE
  {
    /* Réception des autres processeurs */
    OTA_ID(line_anneau, 0, 0, &taille, &taille, 4 ) ;
    wait_l ( line_anneau ) ;
  }
}

```

IV -2.5 Evolution des poids synaptiques

On a choisi une règle qui est celle de Hebb (cf. §II -4.2, Figure 16) donnée par :

$$\Psi_{x_i^k, x_j^k}(w_{ij}(k)) = w_{ij}(k) + f(w_{ij}(k)) \cdot \lambda(x_i(k), x_j(k)) ,$$

$$W_{ij}(k+1) = \begin{cases} \Psi_{x_i^k, x_j^k}(w_{ij}(k)), & \text{Si } j \neq i \\ \Psi_{x_i^k, e_j^{k+1}}(w_{ij}(k)), & \text{Si } j = i \end{cases}$$

- En réalité, $w_{ij}(k) = w_{ij}(k_0), \forall k \in \{k_0, \dots, k_0+K\}; \forall k_0 \geq 1, \text{ où } K = T/\Delta t, T$ est appelée "période d'inertie"

spike à l'instant t sur la synapse W_{ij}	synapse W_{ij}	spike à l'instant t+1 émis par le neurone i	évolution de $ W_{ij} $
1	excitatrice	1	
1	excitatrice	0	
1	inhibitrice	1	
1	inhibitrice	0	
0	excitatrice	1	
0	excitatrice	0	
0	inhibitrice	1	
0	inhibitrice	0	

Figure 16 : les variations des poids synaptiques

• **Implémentation des poids synaptiques excitateurs avec saturation :**

$$W_{ij}(k+1) = \text{Sup}\{\text{Inf}[\Psi_{x_i^k, x_j^{k+1}}(w_{ij}(k)); 1], \epsilon 1\}, k \geq 0$$

avec :

$$\left\{ \begin{array}{l} \lambda(0,1) = -\frac{Sg(w_{ij}(k))}{a.f(w_{ij}(k))} e^{-K}; K > 0 \\ \lambda(0,0) = \lambda(1,0) = -\text{epsilon} \\ \lambda(1,1) = N\text{fois}.\text{epsilon} \end{array} \right.$$

• **Implémentation des poids synaptiques inhibiteurs avec saturation :**

$$W_{il}(k+1) = \text{Inf}\{\text{Sup}[\Psi_{x_i^k, x_l^{k+1}}(w_{il}(k)); -1], \epsilon 2\}, k \geq 0$$

avec :

$$\left\{ \begin{array}{l} \lambda(1,1) = -\frac{Sg(w_{il}(k))}{a.f(w_{il}(k))} e^{-K}; K > 0 \\ \lambda(0,0) = \lambda(1,0) = \text{epsilon} \\ \lambda(0,1) = -N\text{fois}.\text{epsilon} \end{array} \right.$$

où $Sg(x)$ = " signe de x " et où $N\text{fois}$, epsilon et $f(\cdot)$ sont des paramètres de l'équation Hebbienne (cf. §IV -2.9).

On a implémenté trois types de fonctions différentes "f" dans l'équation de Hebb :

$$f(w) = \begin{cases} w & (\text{linéaire}) \\ \frac{w}{w+1} & (\text{homographique}) \\ 1 & (\text{constante}) \end{cases}$$

IV -2.6 Evolution du seuil : φ est prise ici égale à l'identité.

Le seuil évolue comme suit :

pour $k = 0$, $\theta_i(0) = \theta_i^0$ (le seuil initial)

pour $k \geq 1$,

Début

Si $x_i(k) = 1$ **alors** $\theta_i(k) = 1.8$;

Sinon

Début

$\theta_i(k) = \theta_i(k-1) - a.A_i(k-1)$;

Si $\theta_i(k) < 0$ **alors** $\theta_i(k) = 0$;

Sinon { **Si** $\theta_i(k) > 5$ **alors** $\theta_i(k) = 5$ }

Fin

Fin

IV -2.7 Période réfractaire

C'est le temps du repos du neurone. Après l'émission d'un spike par le neurone ($x_i(k) = 1$), il est impossible pour le neurone de décharger durant cette période. On a pris "period_ref=1.2 (ms)"

IV -2.8 Fonction neurone

Dans la pratique, on calcule la sortie " $x_i(k+1)$ " du neurone " i " à l'instant ' $k+1$ ' par deux modes d'évolution (cf. §II -2.4) :

1) Cas déterministe :

$$\begin{cases} X_i(k) = \mathbb{1} \left(\frac{e^{H_i(k)}}{1+e^{H_i(k)}} - \frac{1}{2} \right) \\ \text{où } \mathbb{1}(x) = \begin{cases} 1, & \text{si } x > 0 \\ 0, & \text{si } x \leq 0 \end{cases} \end{cases}$$

2) Cas stochastique :

$$P = \text{Prob} \left\{ X_i(k) = 1 \mid X_j(k-1); j \in N(i), e_i(k) \right\} = \frac{e^{H_i(k)}}{1+e^{H_i(k)}}$$

où $H_i(k)$ est la fonction Hamiltonienne associée au réseau.

Dans le cas stochastique, on tire de façon aléatoire un nombre $\alpha \in [0, 1]$ selon la loi uniforme sur $[0, 1]$:

$$x_i(k) = \begin{cases} 0 & \text{si } \alpha \geq P \\ 1 & \text{si } \alpha < P \end{cases}$$

IV -2.9 Initialisation des paramètres de l'algorithme

- $x_i^0 = 0 ; \forall i \in \Omega$
- *poids synaptiques* :
 - $w_{ii}^0 = 0.75$ (fibre d'entrée)
 - $w_{ij}^0 = 0.50$ ($j \in$ sphère Manhattan 1)
 - $w_{il}^0 = -0.50$ ($l \in$ sphère Manhattan 2)
- Nfois = 16
- $\epsilon_1 = +0.20$
- $\epsilon_2 = - 0.20$
- *Tinertie* = 3 (ms)
- $K = 5$ (cf. § IV 2.5)
- *période réfractaire* = 1.20 (ms)
- *le pas d'échantillonnage* : $\Delta t = 0.01$ ou 0.1 (ms)
- *seuil* : $\theta_i^0(0) = 2.5$

IV -2.10 Récapitulation de l'algorithme du réseau

• Initialisation du réseau

- initialisation des structures de données :
 - (• poids synaptiques : synapses excitatrices, inhibitrices et fibre d'entrée,
 - seuil, refract, Tinertie, period_ref, ...),
- lecture et transfert des paramètres : communication "ota_1d()",
- génération des entrées (génération d'un train inter-spikes),

• Boucle (Nombre d'itérations)

- transfert des états de neurones "coins" : communication "oto_x ()" en 2 étapes,
- parcours de la sous-boîte,
- BOUCLE (sur chaque neurone)
 - test sur la fin d'un intervalle inter-spikes généré :
 - si c'est "oui" alors entree = 1, ensuite générer un nouvel intervalle.
 - si c'est "non" alors entree = 0, ensuite décrémenter le compteur repérant la fin d'intervalle inter-spikes.
 - communications (récupération de l'information voisine)
 - calcul de la fonction d'harmonie (cf. §II -4.2) $A(t) = (A_i(t))_i$
 - calcul de la fonction hamiltonnienne (cf. §II -4.2) $H(t) = (H_i(t))_i$
 - calcul de la sortie (état de sortie du neurone)
 - évolution du seuil
 - évolution des poids synaptiques
 - stockage sur des fichiers binaires "outij.bin" (des configurations codées non nulles)
 - stockage sur des fichiers binaires "innij.bin", dans le cas où on veut répéter les mêmes simulations
- stockage sur des fichiers textes " outij.txt " (des configurations décodées non nulles)
- affichage du temps d'exécution total
- affichage du temps de communication total

IV -3. Réalisations

IV -3.1 Exécutions sur machine parallèle (T-40)

On a observé le réseau pendant 10 ms avec pas d'échantillonnage : $\Delta t=0.01$ (ms), autrement dit, en temps discret, pendant $K_{max}=1000$ itérations.

On définit l'accélération ou le "speed-up" d'une exécution sur une machine à $n=\{4, 16, 32\}$ processeurs comme étant le quotient de T:1proc (temps d'exécution sur un seul processeur) divisé par T:nproc (temps d'exécution sur "n" processeur).

IV -3.1.1 Comparaisons des temps d'exécution de l'algorithme

Taille	T:1proc	T:4proc	T:16proc	T:32proc
8 x 8	1676.651	473.728	142.161	93.025
16 x 16	6712.306	1796.253	475.757	265.722
32 x 32	26475.274	7055.309	1801.067	934.676
64 x 64	104769.086	27797.403	7048.447	3571.929
128 x 128	416421.264	110339.884	27954.440	14040.209
256 x 256	1660002.32	439605.804	111405.717	55753.125

Tableau 1 : Temps d'exécution (secondes)

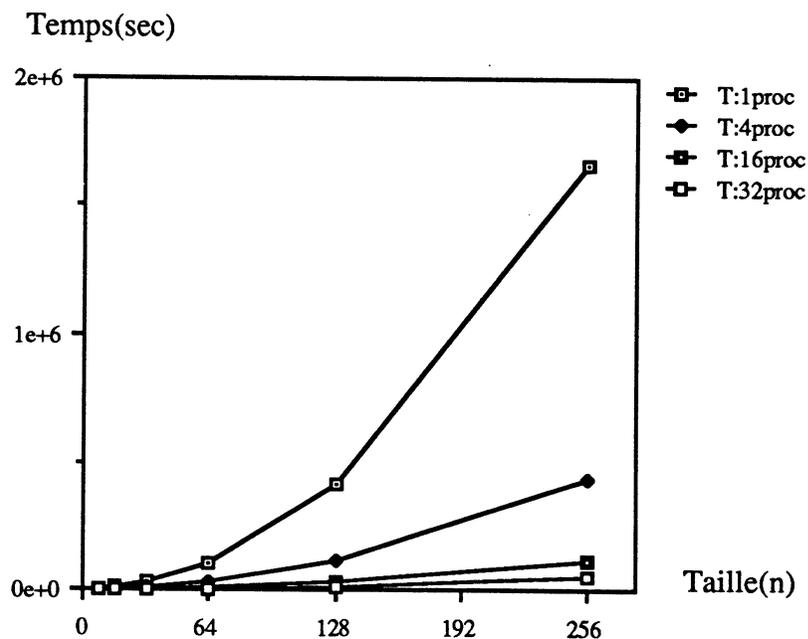


Figure 1 : Temps d'exécution du réseau de taille (nxn)

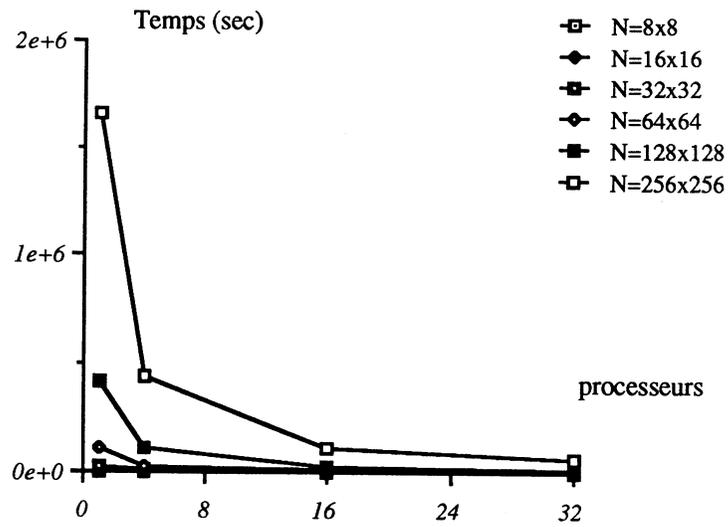


Figure 2 : Temps d'exécution du réseau de taille (N)

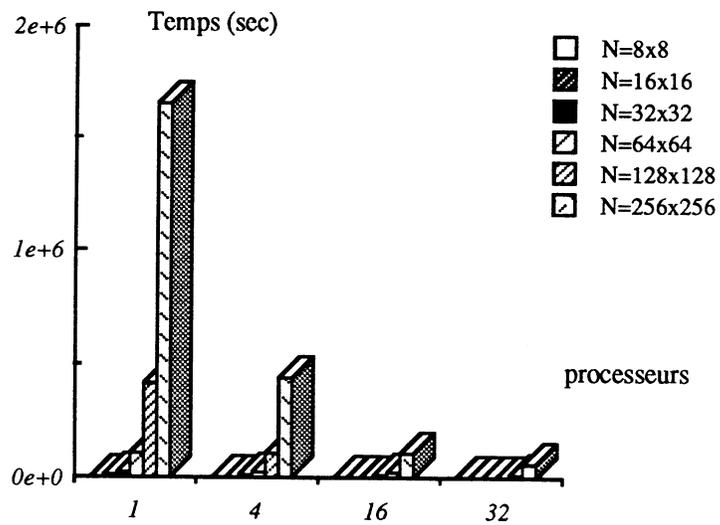


Figure 3 : Temps d'exécution du réseau de taille (N)

Dans les figures 1, 2 et 3, on voit que l'algorithme implémenté sur la machine parallèle de 32 processeurs est deux fois plus rapide que sur celle de 16 processeurs.

IV -3.1.2 Accélérations

Taille	Speed : 4p	Speed : 16p	Speed : 32p
8 x 8	3.539	11.794	18.024
16 x 16	3.737	14.108	25.261
32 x 32	3.752	28.326	
64 x 64	3.777	14.897	29.396
128 x 128	3.774	14.896	29.659
256 x 256	3.776	14.900	29.774

Tableau 2 : Accélération " le speed_up "

Speed_up

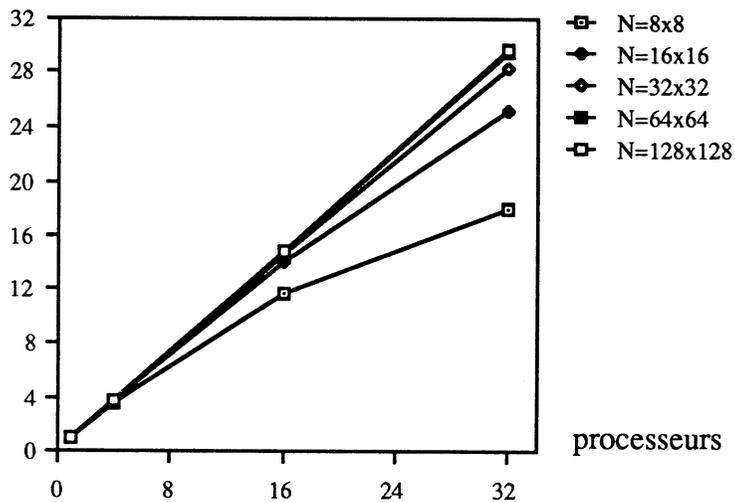


Figure 4 : Accélération du réseau de taille (N)

Speed_up

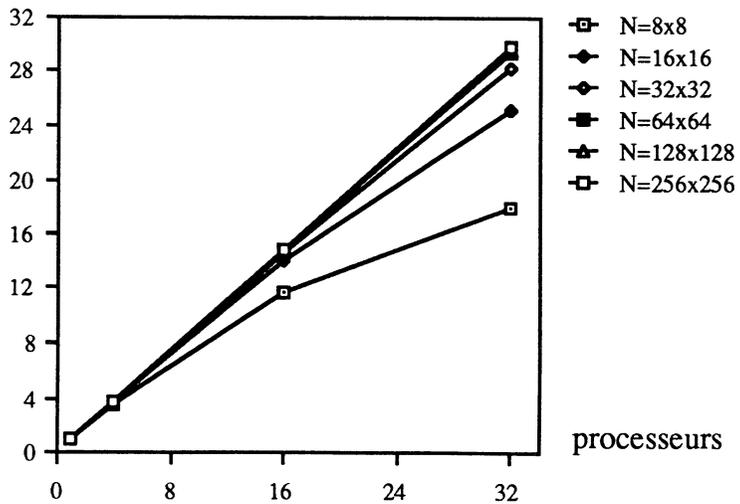


Figure 5 : Accélération du réseau de taille (N)

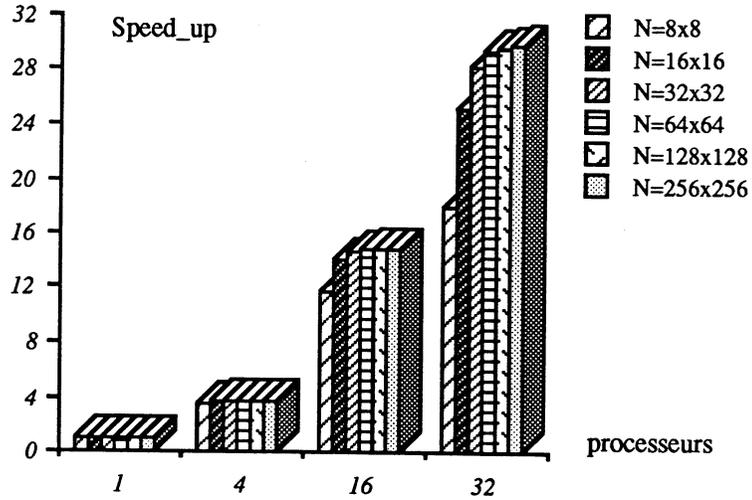


Figure 6 : Accélération du réseau de taille (N)

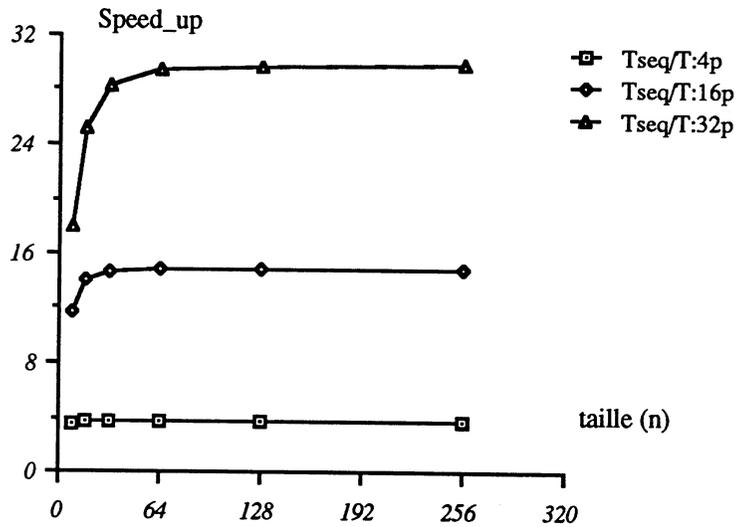


Figure 7 : Accélération du réseau de taille (nxn)

Dans les figures 4 et 5, on voit que le "speed-up" dépend linéairement du nombre de processeurs, dès que la taille du réseau dépasse "32x32" neurones, la droite correspondante se rapproche de la première bissectrice, donc de l'efficacité maximale.

Dans les figures 6 et 7, on voit que le "speed-up" se stabilise, dès que la taille du réseau atteint une valeur seuil, fonction du nombre N de processeurs, ce seuil augmentant de manière quasi-linéaire avec N, et se rapprochant de 8N, lorsque N approche et dépasse 32.

On remarque que l'accélération "speed_up" maximale est donc atteinte pour des réseaux de grande taille. Par exemple, pour le réseau de neurones de taille "256x256", on a obtenu une accélération de "29.774" sur 32 processeurs (cf. Tableau 2).

IV -3.1.3 Comparaison des temps de communication

Taille	Tcom : 4p	Tcom : 16p	Tcom : 32p
8 x 8	38.051	32.341	36.910
16 x 16	52.186	40.266	46.245
32 x 32	80.561	56.920	64.293
64 x 64	191.036	84.978	99.028

Tableau 3 : Temps de communication (secondes)

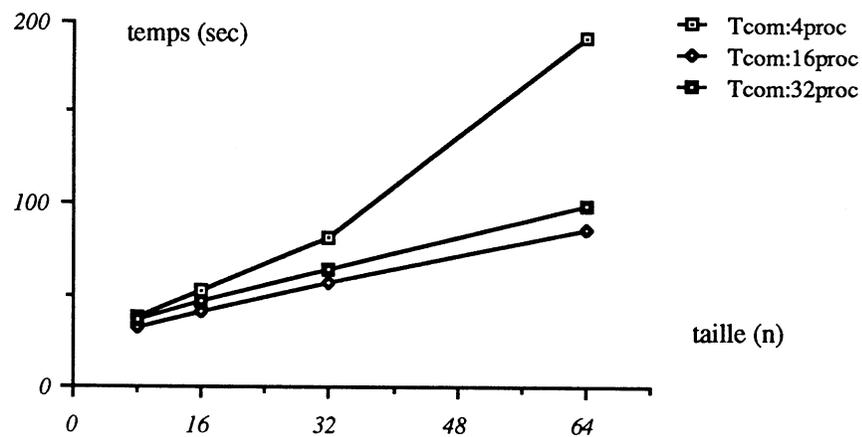


Figure 8 : Temps de communication du réseau de taille (nxn)

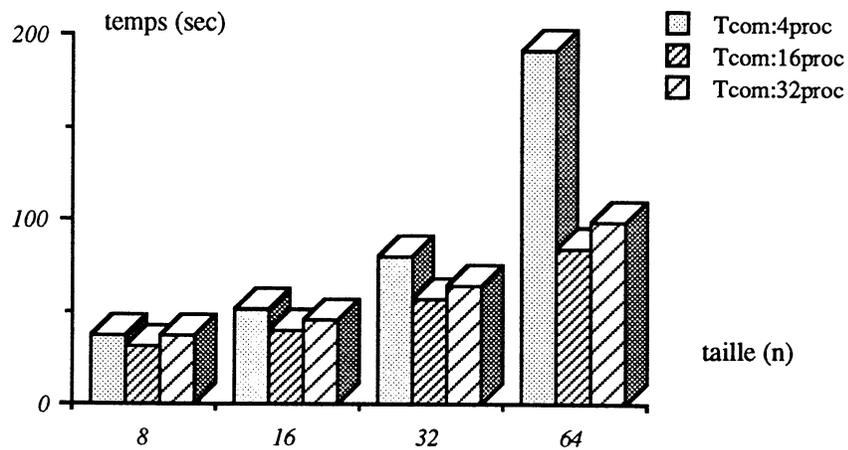


Figure 9 : Temps de communication du réseau de taille (nxn)

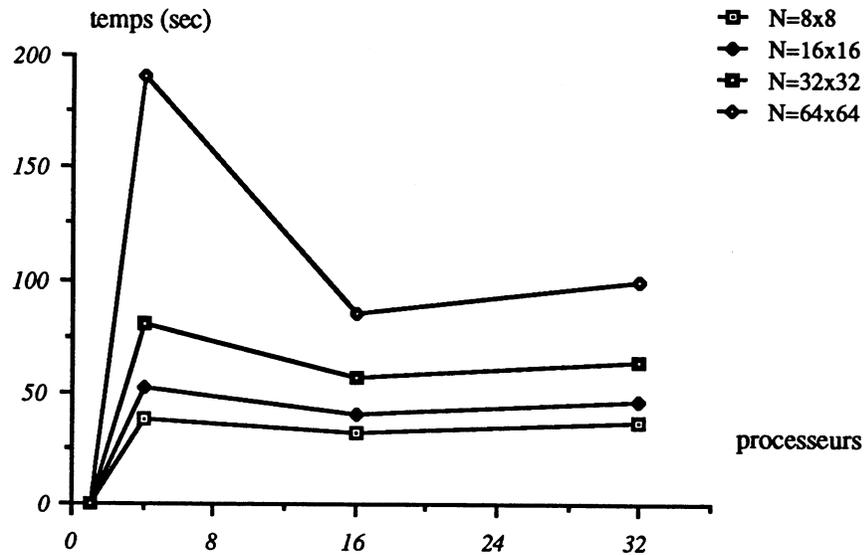


Figure 10 : Temps de communication du réseau de taille (N)

Dans les figures 8 et 9, on voit que le temps de communication croît dès que la taille du réseau croît. Par ailleurs, dans la figure 10, on constate que le temps de communication correspondant à 16 processeurs est inférieur à celui correspondant à 4 processeurs, mais contrairement à ce que l'on pourrait penser, il est aussi inférieur (quoique légèrement) à celui correspondant à 32 processeurs. Cela est dû au multiplexage des liens entre transputers sur un hypercube de 32 processeurs.

IV -3.2 Comparaisons et Résultats de Simulation

IV -3.2.1 Estimations et Tests des lois asymptotiques des configurations

On a observé des réseaux de tailles 8×8 , 16×16 , 32×32 , 64×64 et 128×128 , en différents modes d'implémentation (parallèle et bloc-séquentiel), pendant 10200 itérations, après le temps de convergence. On note "*n-uplet*" la configuration où "*n*" neurones sont chargés (ou excités) en même temps (cf. tableaux et graphiques).

Dans le premier paragraphe, on a représenté les histogrammes correspondants aux évolutions des réseaux de taille et mode d'implémentation différents.

Dans le second paragraphe, on a estimé ces lois d'évolutions par des fonctionnelles, dites : "*fonctions à noyaux*" (cf. Khelladi et al. (87), A. Berlinet(86) et Annexe 1). Ensuite, on les a testées contre des lois usuelles (type : Poisson, Binomiale, Gauss, ...), par un test, appelé : "*test de Khi-deux*" (cf. Annexe 1). D'après les tests effectués, on a

toujours rejeté l'hypothèse que "la loi observée est la loi théorique". Sauf si l'on compare les lois tronquées en 0, auquel cas on obtient une adéquation à la loi de Poisson conditionnelle à l'observation non vide. Pour cela, on a préféré présenter dans ce manuscrit une seule illustration, numérique et graphique, dans le cas du réseau de taille "32x32" en modes parallèle et bloc-séquentiel, les deux modes donnant d'ailleurs des résultats voisins jusqu'à la taille "128x128" pour laquelle ils commencent à avoir des lois différentes. Cette différence ne fait que s'affirmer lorsque la taille augmente (*cf. surtout Figure 26*). Nous observons, sur simulation numérique, un résultat du même type que celui démontré dans un cas simple pp. 40-42 ci-dessus ; nous reviendrons sur ces différences p. 162, en montrant pourquoi les itérations parallèles chargent plus les amas de petites ou de grandes tailles, les itérations bloc-séquentielles privilégiant les configurations comportant des amas de taille moyenne.

Envisageons d'abord les histogrammes d'évolution des réseaux de neurones : nous partirons en abscisse la taille n d'un n -uplet et en ordonnée la fréquence avec laquelle a été observé de manière isolée l'occurrence d'un n -uplet, c'est-à-dire la fréquence de la taille de la configuration observée.

Nous nous intéressons :

- 1) aux différences existant entre modes de mise à jour parallèle et bloc-séquentiel.
- 2) aux lois classiques (Poisson, etc...) susceptibles d'être sous-jacentes aux observations.

C'est pourquoi nous donnerons successivement systématiquement les histogrammes parallèle et bloc-séquentiel, et nous ferons des tests d'adéquation de ces histogrammes à toutes les lois de distribution classiques.

1° - Histogrammes des lois asymptotiques des configurations des réseaux

• Réseau à comportement parallèle de taille (N)

n-uplets	nb_apparition
0	7320
1	2422
2	406
3	48
4	3
5	0

Tableau 4

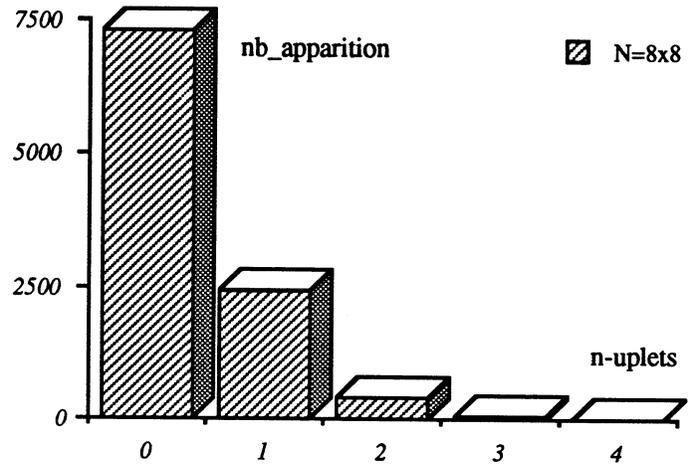


Figure 12

• Réseau à comportement bloc-séquentiel de taille (N)

n-uplets	nb_apparition
0	7330
1	2141
2	404
3	42
4	6
5	1
6	0

Tableau 5

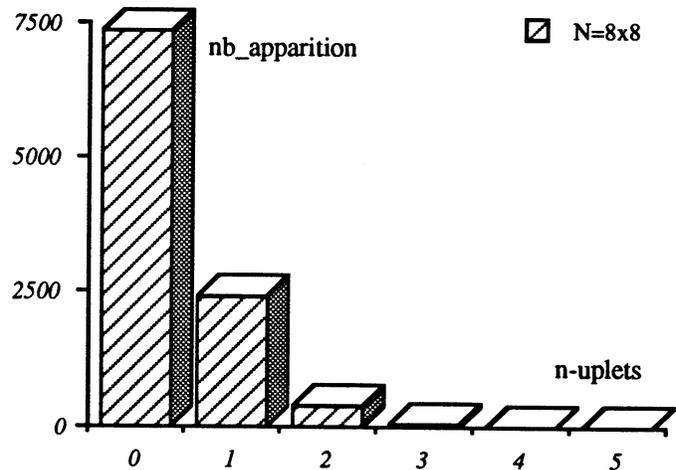


Figure 13

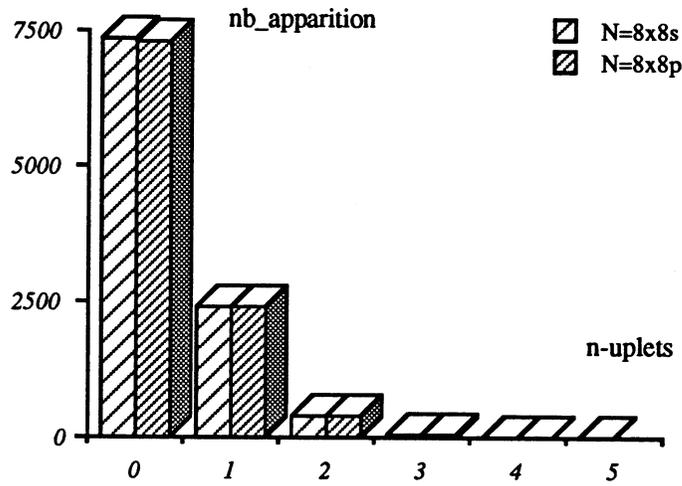


Figure 14 : Courbes simultanées (12 et 13)

• Réseau à comportement parallèle de taille (N)

n-uplets	nb_apparition
0	2850
1	3763
2	2606
3	1096
4	371
5	84
6	22
7	7
8	1
9	0

Tableau 6

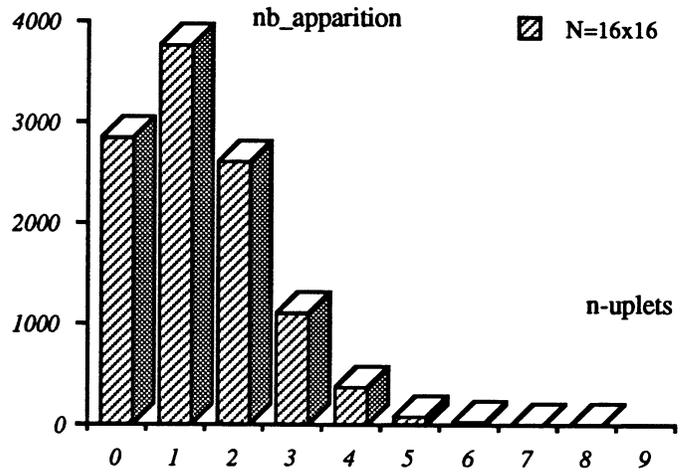


Figure 15

• Réseau à comportement bloc – séquentiel de taille (N)

n-uplets	nb_apparition
0	2757
1	3558
2	2300
3	1101
4	364
5	100
6	17
7	3
8	0

Tableau 7

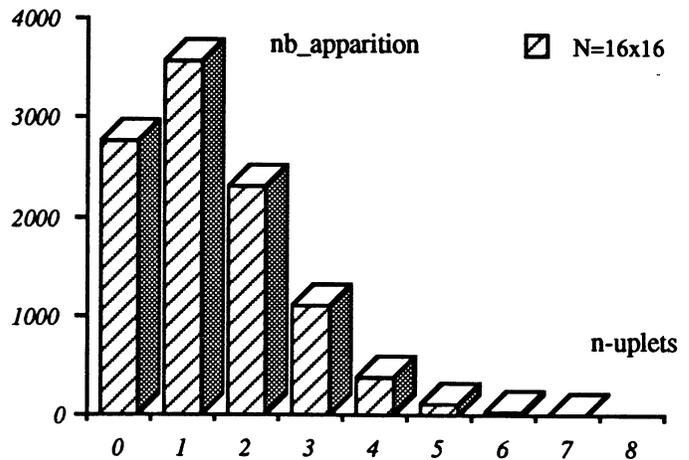


Figure 16

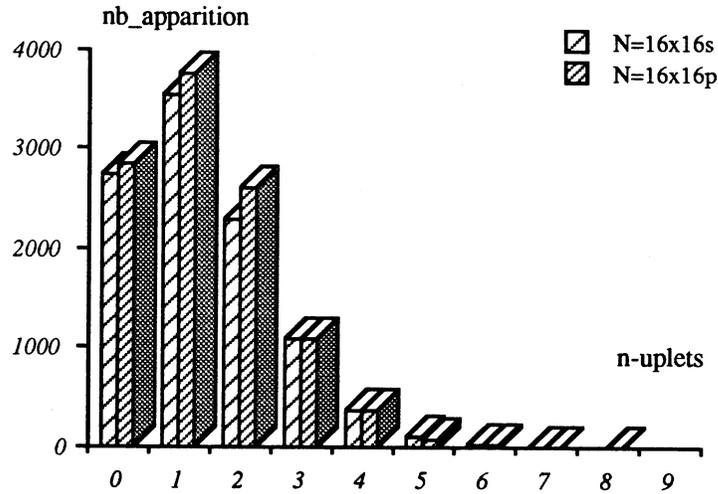


Figure 17 : Courbes simultanées (15 et 16)

• Réseau à comportement parallèle de taille (N)

n-uplets	nb_apparition
0	86
1	300
2	712
3	1188
4	1715
5	1732
6	1516
7	1223
8	805
9	449
10	270
11	116
12	50
13	25
14	5
15	6

Tableau 8

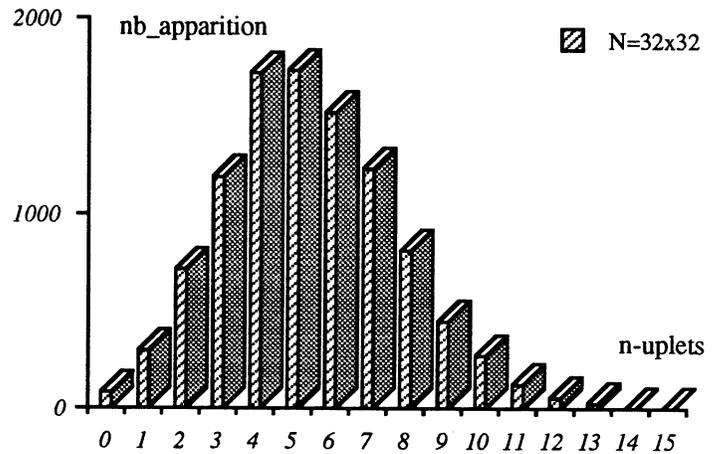


Figure 18

• Réseau à comportement bloc-séquentiel de taille (N)

n-uplets	nb_apparition
0	94
1	280
2	706
3	1241
4	1616
5	1746
6	1529
7	1202
8	817
9	495
10	256
11	118
12	68
13	24
14	7
15	1

Tableau 9

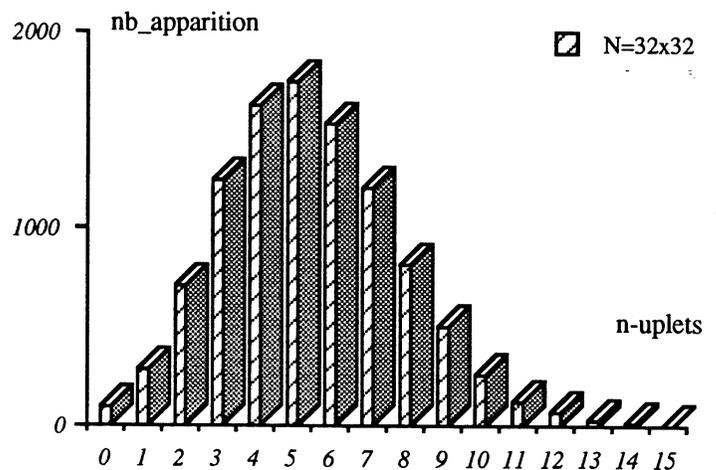


Figure 19

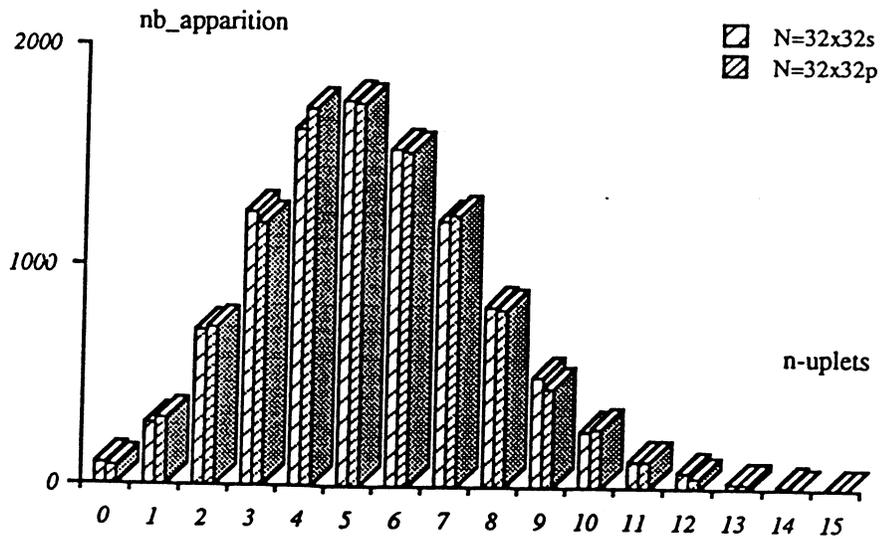


Figure 20 : Courbes simultanées(18 et 19)

X	Y	X	Y	X	Y	X	Y
0	22	11	74	22	780	33	62
1	4	12	124	23	796	34	36
2	6	13	198	24	673	35	31
3	6	14	295	25	596	36	15
4	3	15	381	26	484	37	8
5	8	16	466	27	385	38	2
6	6	17	581	28	350	39	1
7	17	18	676	29	220	40	0
8	18	19	754	30	173	41	0
9	30	20	843	31	111	42	1
10	36	21	848	32	80	43	0

Tableau 10 : Réseau en mode parallèle de taille $N = 64 \times 64$

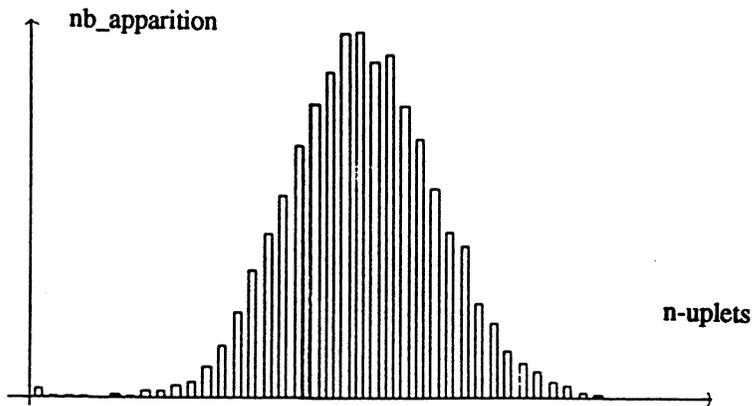


Figure 21 : Réseau en mode parallèle taille (N)

X	Y	X	Y	X	Y	X	Y
0	26	11	75	22	809	33	50
1	1	12	116	23	769	34	40
2	5	13	193	24	673	35	28
3	5	14	260	25	612	36	16
4	5	15	388	26	449	37	9
5	8	16	492	27	392	38	4
6	7	17	583	28	302	39	1
7	11	18	713	29	247	40	3
8	15	19	776	30	183	41	1
9	31	20	818	31	117	42	3
10	66	21	821	32	75	43	1

Tableau 11 : Réseau en mode bloc-séquentiel de taille $N = 64 \times 64$

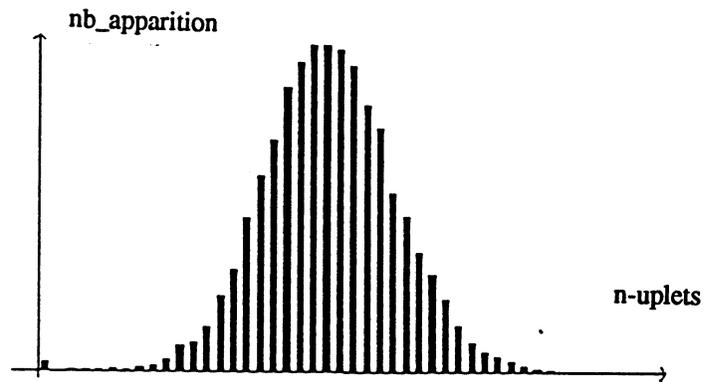


Figure 22 : Réseau en mode bloc-séquentiel de taille (N)

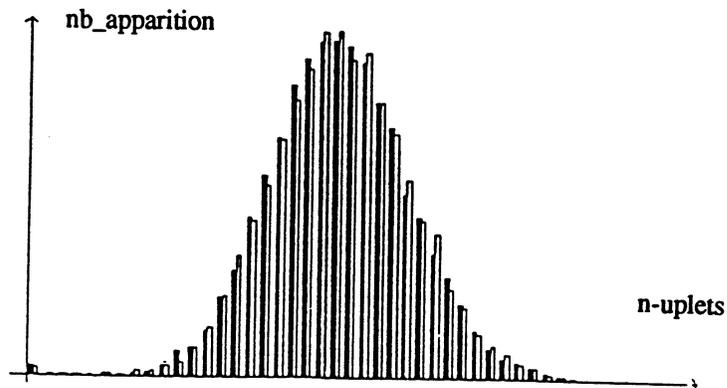


Figure 23 : Courbes simultanées 21 et 22

X	Y	X	Y	X	Y	X	Y	X	Y
0	26	25	1	50	5	75	231	100	157
1	2	26	3	51	7	76	250	101	145
2	2	27	2	52	8	77	227	102	117
3	1	28	0	53	11	78	260	103	102
4	1	29	0	54	13	79	309	104	97
5	0	30	0	55	14	80	308	105	67
6	2	31	1	56	20	81	349	106	61
7	1	32	3	57	21	82	344	107	47
8	2	33	0	58	21	83	356	108	51
9	1	34	0	59	28	84	348	109	30
10	1	35	2	60	32	85	359	110	24
11	1	36	1	61	40	86	410	111	17
12	0	37	4	62	38	87	382	112	19
13	1	38	4	63	49	88	378	113	17
14	1	39	2	64	57	89	356	114	7
15	1	40	2	65	65	90	366	115	9
16	0	41	2	66	70	91	373	116	4
17	0	42	3	67	63	92	311	117	5
18	0	43	2	68	97	93	291	118	3
19	0	44	5	69	106	94	285	119	1
20	1	45	2	70	144	95	254	120	2
21	0	46	7	71	146	96	227	121	2
22	3	47	0	72	149	97	208	122	1
23	1	48	7	73	185	98	193	123	2
24	2	49	6	74	170	99	173	124	1

Tableau 12 : Réseau en mode parallèle de taille $N = 128 \times 128$

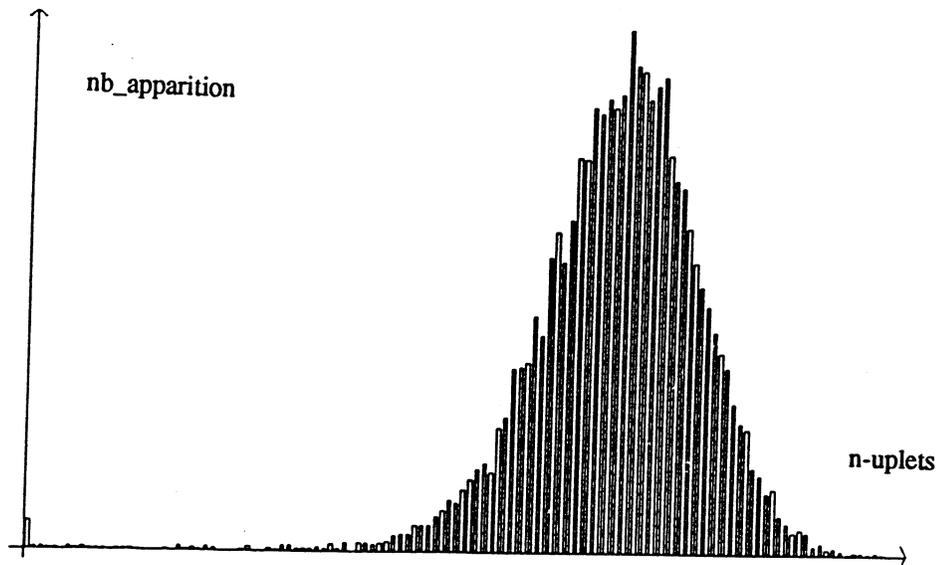


Figure 24 : Réseau en mode parallèle taille (N)

X	Y	X	Y	X	Y	X	Y	X	Y
0	22	25	0	50	4	75	282	100	117
1	2	26	2	51	7	76	314	101	86
2	0	27	0	52	11	77	308	102	79
3	3	28	4	53	11	78	316	103	75
4	1	29	1	54	9	79	325	104	47
5	1	30	1	55	14	80	372	105	63
6	0	31	1	56	23	81	341	106	38
7	1	32	2	57	19	82	378	107	33
8	1	33	0	58	33	83	361	108	29
9	1	34	2	59	45	84	335	109	20
10	1	35	3	60	44	85	355	110	12
11	0	36	0	61	39	86	363	111	17
12	0	37	5	62	47	87	344	112	5
13	1	38	4	63	84	88	319	113	7
14	0	39	0	64	86	89	308	114	5
15	2	40	0	65	82	90	304	115	7
16	1	41	3	66	15	91	302	116	3
17	0	42	1	67	126	92	274	117	3
18	2	43	1	68	144	93	241	118	2
19	2	44	1	69	163	94	226	119	4
20	0	45	1	70	181	95	219	120	0
21	0	46	2	71	213	96	183	121	0
22	1	47	3	72	203	97	170	122	1
23	2	48	8	73	247	98	158	123	0
24	3	49	8	74	248	99	149	124	2

Tableau 13 : Réseau en mode bloc-séquentiel de taille $N = 128 \times 128$

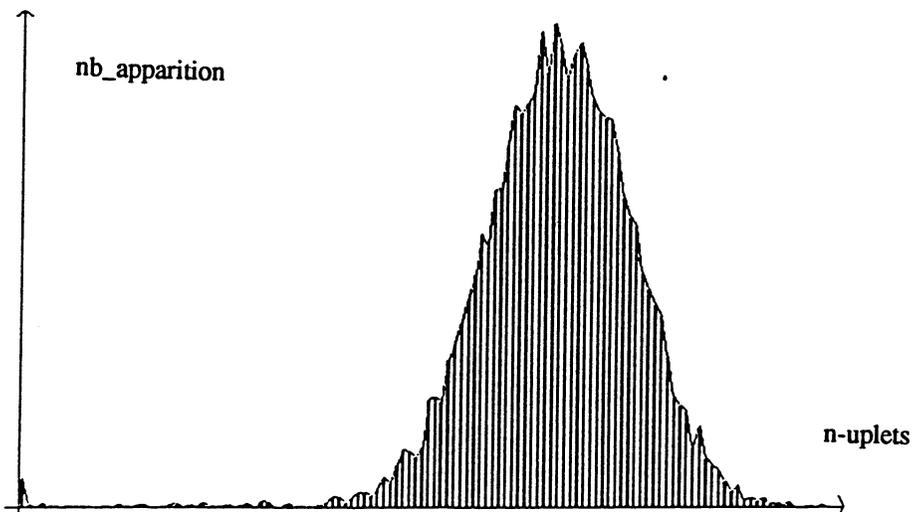


Figure 25 : Réseau en mode bloc-séquentiel de taille (N)

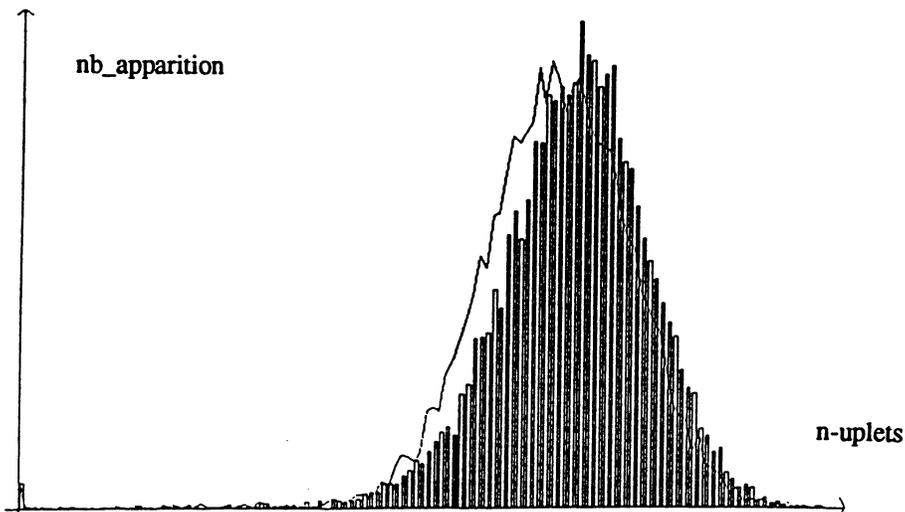


Figure 26 : Courbes simultanées 24 et 25

La conclusion de l'observation des figures 12 à 26 est que l'histogramme des fréquences des tailles des configurations observées est moins dispersé en mode bloc-séquentiel qu'en mode parallèle et que le maximum de fréquence en mode bloc-séquentiel est atteint pour une taille inférieure à celui du mode parallèle.

Nous commenterons cette observation en proposant une explication (pp. 162) du fait que les fréquences des configurations de taille intermédiaire sont plus importantes dans le cas bloc-séquentiel que dans le cas parallèle et que le mode parallèle charge plus de configurations de taille faible ou grande.

Dans la Figure 26, on remarque que la courbe des fréquences, dans le cas parallèle, est décalée vers la droite par rapport à celle dans le cas bloc-séquentiel. Ce qui explique la différence entre ces deux échantillons. Nous validons cette différence par un test statistique significatif en comparant les deux moyennes correspondants comme suit :

Test des moyennes :

Soit Moy_{par} (resp. Moy_{seq}) la moyenne de l'échantillon en mode parallèle (resp. moyenne de l'échantillon en mode bloc-séquentiel) et de même pour Var_{par} et Var_{seq} (les variances associées).

• Le test consiste à tester l'hypothèse H_0 : “ $Moy_{par} = Moy_{seq}$ ” contre H_1 : “ $Moy_{par} \neq Moy_{seq}$ ” par un test des moyennes en calculant la statistique T telle que :

$$T = \frac{Moy_{par} - Moy_{seq}}{\sqrt{\frac{Var_{par}}{N_{par}} + \frac{Var_{seq}}{N_{seq}}}} \rightarrow N(0, 1)$$

où N_{par} et N_{seq} sont des tailles d'échantillon.

et on a : $Prob \{ |N(0, 1)| > 1.96 \} = \alpha = 5\%$, la probabilité de rejeter l'hypothèse H_0 à un risque de 1^{ère} espèce α .

Dans notre cas :	$Moy_{par} = 84.82$	$Moy_{seq} = 82.52$
	$Var_{par} = 163.02$	$Var_{seq} = 155.84$
	$N_{par} = 10200$	$N_{seq} = 10200$

alors on a $T = 13.03 > 1.96$, par conséquent on rejette l'hypothèse H_0 .

n-uplets	sortie	entrée
0	101	6567
1	366	4349
2	1008	2604
3	1773	1028
4	2364	335
5	2523	87
6	2321	23
7	1840	4
8	1203	2
9	771	0
10	398	0
11	182	0
12	94	0
13	36	0
14	18	0
15	2	0
16	0	0

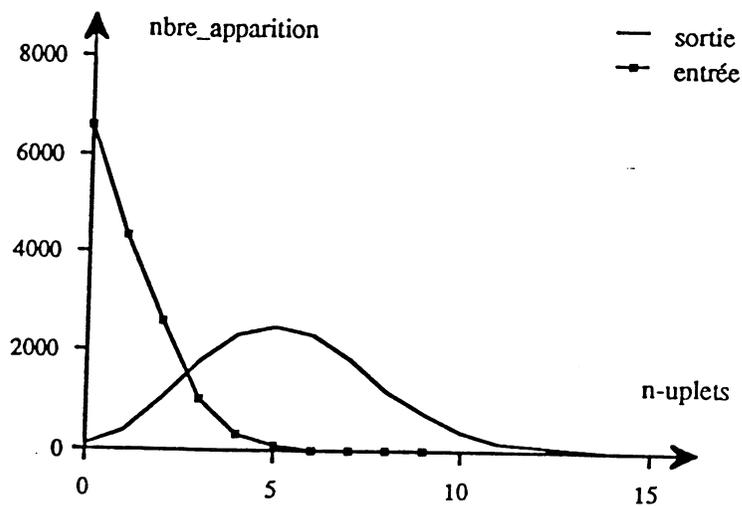


Figure 27 : Réseau de neurones de taille $N=32 \times 32$ en mode d'implémentation parallèle (ou synchrone). Le réseau est observé pendant 15000 itérations (ou 150 millisecondes pour un pas d'échantillonnage $\Delta t=0.01$ ms). L'entrée (ou l'innovation extérieure) est non homogène en temps, c'est-à-dire que les neurones du réseau reçoivent des entrées qui sont générées par deux lois de densité de probabilité de Gauss $N(m, \sigma^2)$ de moyenne "m" et de variance " σ^2 ", telles que : pendant les premières 9000 itérations, $m=8$ et $\sigma^2=2$ et, pendant les autres 6000 itérations, $m=20$ et $\sigma^2=2$ (cf. § IV -2.3.1).

n-uplets	sortie	entrée
0	101	4216
1	376	5274
2	994	3449
3	1764	1427
4	2380	472
5	2551	122
6	2307	34
7	1813	4
8	1237	1
9	734	0
10	377	0
11	205	0
12	94	0
13	42	0
14	18	0
15	5	0
16	0	0
17	0	0

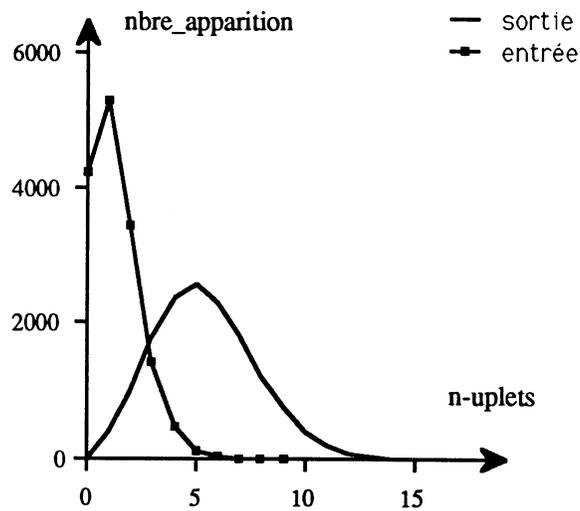


Figure 28 : Réseau de neurones de taille $N=32 \times 32$ en mode d'implémentation parallèle (ou synchrone). Le réseau est observé pendant 15000 itérations (ou 150 ms pour un pas d'échantillonnage $\Delta t = 0.01$ ms). L'entrée (ou l'innovation extérieure) est homogène, c'est-à-dire que les neurones du réseau reçoivent des entrées générées par une même loi de densité de probabilité de Gauss $N(m, \sigma^2)$ de moyenne "m" et de variance " σ^2 " telles que : pendant toute la période d'observation , $m=8$ et $\sigma^2=2$ (cf. § IV -2.3.1).

n-uplets	sortie-par	sortie-bseq	entrée
0	91	97	5115
1	405	364	5408
2	1002	1038	2981
3	1709	1768	1120
4	2346	2389	299
5	2620	2555	61
6	2344	2275	11
7	1811	1800	3
8	1189	1247	1
9	737	757	0
10	394	384	0
11	209	198	0
12	83	80	0
13	43	29	0
14	14	13	0
15	3	6	0
16	0	0	0

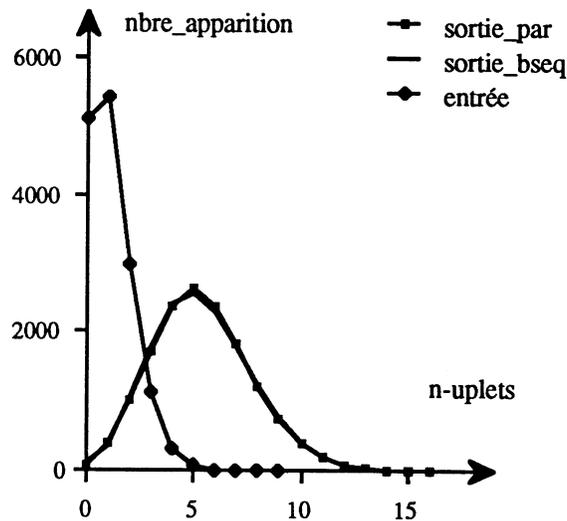


Figure 29 : Réseau de neurones de taille $N=32 \times 32$ en modes d'implémentations parallèle (*sortie_par*) et bloc-séquentiel (*sortie_bseq*). Le réseau est observé pendant 15000 itérations (ou 150 ms pour un pas d'échantillonnage $\Delta t=0.01$ ms). L'entrée (ou l'innovation extérieure) est non homogène en espace, c'est-à-dire que les neurones du réseau reçoivent des entrées qui sont générées par des lois différentes. Dans notre cas, les trois premiers quarts du réseau reçoivent des entrées générées par la densité de probabilité de Gauss $N(m, \sigma^2)$ de moyenne " $m=8$ " et de variance " $\sigma^2=2$ ", et dans l'autre quart du réseau, elles sont générées par la même famille de lois, mais avec une moyenne " $m=20$ " et une variance " $\sigma^2=2$." (cf § IV -2.3.1).

n-uplets	sortie-Wcst	sortie-Wvar	entrée
0	133	101	9361
1	492	385	4053
2	1217	1070	1200
3	2099	1781	312
4	2597	2417	57
5	2656	2516	17
6	2071	2317	0
7	1554	1741	0
8	1049	1242	0
9	602	695	0
10	276	412	0
11	154	180	0
12	59	87	0
13	43	34	0
14	30	14	0
15	7	6	0
16	4	2	0
17	0	0	0

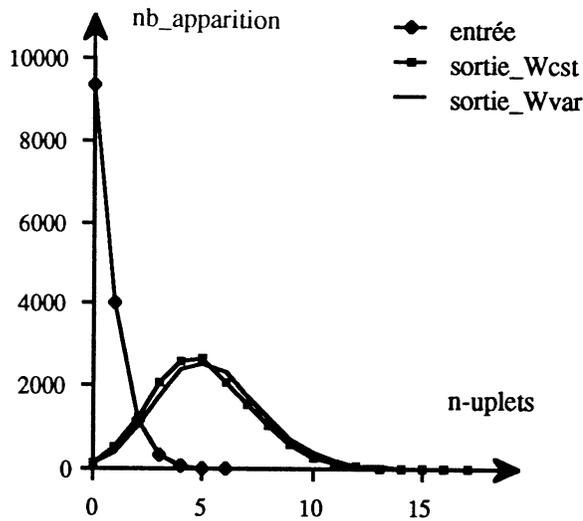


Figure 30 : Réseau de neurones de taille $N=32 \times 32$ en mode d'implémentation parallèle (ou synchrone). Le réseau est observé pendant 15000 itérations (ou 150 ms pour un pas d'échantillonnage $\Delta t = 0.01$ ms). L'entrée (ou l'innovation extérieure) est homogène, c'est-à-dire que les neurones du réseau reçoivent des entrées générées par une même loi de densité de probabilité de Gauss $N(m, \sigma^2)$ de moyenne " $m=20$ " et de variance " $\sigma^2=2$ ", (cf. § IV -2.3.1). Pendant toute la période d'observation, d'une part, on a laissé les poids synaptiques constants (sortie_Wcst) et d'autre part, on a pris des poids synaptiques variables selon la règle de Hebb (cf. § IV -2.5).

Conclusions :

a) le "transfert" non linéaire dû au réseau donne une variance de sortie d'autant plus grande que la variance d'entrée l'est (*Figures 29 et 30*), ce qui était un résultat attendu (comparable à ce qui s'observe pour les réseaux réels).

b) sur un réseau 32×32 , la variabilité des poids a plus d'influence que le mode d'implémentation (*Figures 29 et 30*), ce qui n'était pas évident au départ et restait à montrer.

dans les pages qui suivent (pp. 106-126), nous allons tester systématiquement l'adéquation des histogrammes des réseaux 32×32 (en mode parallèle, puis en bloc-séquentiel) à des lois de distribution classiques : Poisson, binomiale, Gauss, Gamma, khi-deux et Weibull.

Le seul test où l'on peut accepter avec un seuil de significativité correct (1%) l'adéquation est celui (pp. 126) à une loi de Weibull, à condition de supprimer l'occurrence de la configuration vide (taille 0). L'analogie avec un phénomène de vieillissement (modélisé en générale par une loi du Weibull) est difficile à formuler, car il n'y a pas d'apparente relation entre les tailles des configurations observées et les durées de vie d'un système aléatoire associé. Le transfert réalisé entre une loi renouvelante géométrique en entrée et une loi de Weibull en sortie peut peut-être s'expliquer par l'intégration spatiale d'une loi temporelle sans mémoire.

2°- Tests des lois d'évolution des réseaux de neurones

* Test contre une loi de Poisson :

TEST D'HYPOTHESE DU KHI-DEUX

Réseau de neurones de taille (taille) ----> 32 x 32
 Nombre total d'itérations (Kmax) ----> 10200

Type de mode d'implémentation du réseau :
 mode parallèle --> note par 'par'
 mode block-séquentiel --> note par 'bseq'
 Votre choix exact ----> par

Entrez le nombre total d'observations (Np) ----> 20
 Entrez le coefficient de correction du pas H0(coef)----> 2.30

Entrez le type de densité théorique :
 Densité Normale(m, sigma) --> Entrez 'n'
 Densité Gamma(a,b) --> Entrez 'g'
 Densité Khi-deux a n d*1 --> Entrez 'k'
 Densité Weibull(m,a,b) --> Entrez 'w'
 Densité de Poisson --> Entrez 'p'
 Densité Binomiale --> Entrez 'b'
 Votre choix ----> p.

Voulez-vous ouvrir le fichier densité à noyau(o/n) ----> n
 Voulez-vous ouvrir le fichier densité théorique(o/n) ----> o
 Nom du fichier lecture ----> 32pkhi2.dat
 Nom du fichier écriture pour densité théorique ----> 32ppoisth.dat

- * moyenne arithmétique = 5.2981
- * moyenne géométrique = 4.7271
- * SQRT(variance empirique) = 2.3436
- * Le pas optimal de la densité à noyau: H0 = 0.1472
- * Le pas utilisé : H0 = H0*coef (optimal) = 0.3385
- * la somme des Tnoy[i] : some = 1.0098
- * la loi théorique est la loi Poisson(5.30)

xi	ni	pi	npi	khi2i
0	86	0.0050	51.0092	24.0027
1	300	0.0265	270.2537	3.2741
2	712	0.0702	715.9205	0.0215
3	1188	0.1240	1264.3484	4.6103
4	1715	0.1642	1674.6728	0.9711
5	1732	0.1740	1774.5293	1.0193
6	1516	0.1536	1566.9500	1.6567
7	1223	0.1163	1185.9880	1.1551
8	805	0.0770	785.4409	0.4871
9	449	0.0453	462.3749	0.3869
10	270	0.0240	244.9725	2.5569
11	116	0.0116	117.9907	0.0336
12	50	0.0051	52.0943	0.0842
13	25	0.0021	21.2310	0.6691
14	5	0.0008	8.0346	0.0504
15	6	0.0003	2.8379	--
16	1	0.0001	0.9397	--
17	0	0.0000	0.2929	--
18	1	0.0000	0.0862	--
19	0	0.0000	0.0240	--
sum	10200	1		40.9789

- * la statistique du Khi-deux calculée = 40.9789
- * le degré de liberté de khi-deux = 13

Decision 1 :

$\forall \alpha > 0.001$ (le risque de 1ère espèce) tel que $P [\chi^2 (13) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Poisson (5.30) (cf. Figure 31).

* la loi théorique est la loi Poisson (5.34)

xi	ni	pi	npi	khi2i
1	300	0.0255	258.3488	6.7150
2	712	0.0682	690.2031	0.6884
3	1188	0.1215	1229.2949	1.3872
4	1715	0.1624	1642.0883	3.2374
5	1732	0.1735	1754.7972	0.2962
6	1516	0.1545	1562.7018	1.3957
7	1223	0.1179	1192.8298	0.7631
8	805	0.0788	796.6892	0.0867
9	449	0.0468	472.9844	1.2162
10	270	0.0250	252.7245	1.1809
11	116	0.0121	122.7595	0.3722
12	50	0.0054	54.6606	0.3974
13	25	0.0022	22.4663	0.2857
14	5	0.0008	8.5744	0.0006
15	6	0.0003	3.0543	--
16	1	0.0001	1.0200	--
17	0	0.0000	0.3206	--
18	1	0.0000	0.0952	--
19	0	0.0000	0.0268	--
som	10114	1		18.0227

* la statistique du Khi-deux calculée = 18.0227
 * le degré de liberté de khi-deux = 12

Decision 2 :

$\forall \alpha \leq 10\%$ (le risque de 1^{ère} espèce) tel que $P [\chi^2(12) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T < A_\alpha$. Donc, on accepte l'hypothèse que la loi soit une loi de Poisson (5.34) (cf. Figure 31).

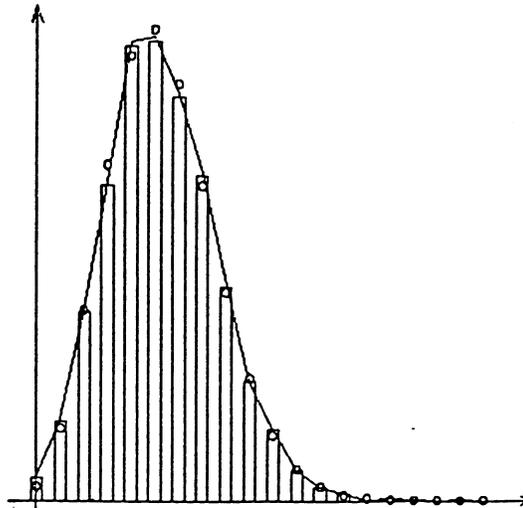


Figure 31 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Poisson (P[5.30]) en symboles ronds dont le paramètre est estimé à partir de l'échantillon. Dans le tableau juste au-dessus, on a supprimé la première observation " 0-uplet " (la configuration vide), ce qui a induit une acceptation de l'hypothèse de Poisson.

TEST D'HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations         ( Kmax)   ---> 10200

Type de mode d'implémentation du réseau :
mode parallèle                    --> note par 'par'
mode block-sequential             --> note par 'bseq'
Votre choix exact                  ---> bseq

Entrez le nombre total d'observations (Np)      ---> 17
Entrez le coefficient de correction du pas H0(coef) ---> 2.30

Entrez le type de densité théorique :
Densité Normale(m, sigma)         --> Entrez 'n'
Densité Gamma(a,b)                --> Entrez 'g'
Densité Khi-deux a n d'l          --> Entrez 'k'
Densité Weibull(m,a,b)            --> Entrez 'w'
Densité de Poisson                --> Entrez 'p'
Densité Binomiale                 --> Entrez 'b'
Votre choix                        ---> p

Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> n
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture              ---> 32skhi2.dat
Nom du fichier écriture pour densité théorique ---> 32spoisth.dat

* moyenne arithmétique           = 5.3223
* moyenne géométrique            = 4.7496
* SQRT(variance empirique)       = 2.3534

* Le pas optimal de la densité a noyau: H0 = 0.1476
* Le pas utilise : H0 = H0*coef (optimal) = 0.3394
* la somme des Tnoy[i] :          some = 1.0072
    
```

* la loi théorique est la loi Poisson(5.32)

xi	ni	pi	npi	khi2i
0	94	0.0049	49.7937	39.2459
1	280	0.0260	265.0147	0.8474
2	706	0.0691	705.2378	0.0008
3	1241	0.1227	1251.1518	0.0824
4	1616	0.1632	1664.7372	1.4268
5	1746	0.1737	1772.0311	0.3824
6	1529	0.1541	1571.8669	1.1690
7	1202	0.1172	1195.1252	0.0395
8	817	0.0780	795.0951	0.6035
9	495	0.0461	470.1888	1.3093
10	256	0.0245	250.2464	0.1323
11	118	0.0119	121.0796	0.0783
12	68	0.0053	53.7014	3.8072
13	24	0.0022	21.9856	0.1846
14	7	0.0008	8.3581	12.3101
15	1	0.0003	2.9656	--
16	0	0.0001	0.9865	--
scm	10200	1		50.8185

* la statistique du Khi-deux calculée = 50.8185
 * le degré de liberté de khi-deux = 13

Decision 3 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (13) > A \alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A \alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Poisson (5.32) (cf. Figure 32).

* la loi théorique est la loi Poisson (5.37)

xi	ni	pi	npi	khi2i
1	280	0.0250	252.2147	3.0610
2	706	0.0670	677.4183	1.2059
3	1241	0.1200	1212.9761	0.6474
4	1616	0.1612	1628.9539	0.1030
5	1746	0.1732	1750.0697	0.0095
6	1529	0.1550	1566.8255	0.9132
7	1202	0.1190	1202.3728	0.0001
8	817	0.0799	807.3572	0.1152
9	495	0.0477	481.8809	0.3572
10	256	0.0256	258.8548	0.0315
11	118	0.0125	126.4096	0.5595
12	68	0.0056	56.5868	2.3020
13	24	0.0023	23.3824	0.0163
14	7	0.0009	8.9718	13.2634
15	1	0.0003	3.2129	2.0887
16	0	0.0001	1.0787	--
som	10106	1		11.4103

* la statistique du Khi-deux calculée = 11.4103
 * le degré de liberté de khi-deux = 12

Decision 4 :

$\forall \alpha \leq 50\%$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (12) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T < A_\alpha$. Donc, on accepte l'hypothèse que la loi soit une loi de Poisson (5.37) (cf. Figure 32).

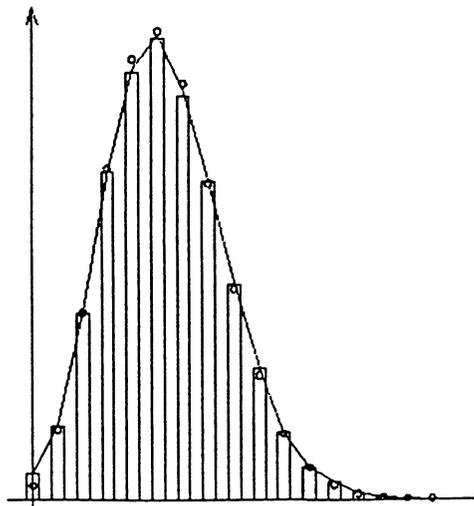


Figure 32 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Poisson ($P[5.32]$) en symboles ronds dont le paramètre est estimé à partir de l'échantillon. Dans le tableau juste au-dessus, on a supprimé la première observation " 0-uplet " (la configuration vide), ce qui a induit une acceptation de l'hypothèse de Poisson.

*** Test contre une loi Binomiale :**

TEST D'HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations         ( Kmax)    ---> 10200

Type de mode d'implémentation du réseau :
mode parallèle      --> note par 'par'
mode block-sequential --> note par 'bseq'
Votre choix exact          ---> par

Entrez le nombre total d'observations (Np)      ---> 20
Entrez le coefficient de correction du pas H0(coef) ---> 2.30

Entrez le type de densité théorique :
Densité Normale(m, sigma)  --> Entrez 'n'
Densité Gamma(a,b)        --> Entrez 'g'
Densité Khi-deux a n d'1   --> Entrez 'k'
Densité Weibull(m,a,b)    --> Entrez 'w'
Densité de Poisson        --> Entrez 'p'
Densité Binomiale         --> Entrez 'b'
Votre choix                ---> b

Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> n
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture      ---> 32pkhi2.dat
Nom du fichier écriture pour densité théorique ---> 32pbinoth.dat

* moyenne arithmétique      = 5.2981
* moyenne géométrique      = 4.7271
* SQRT(variance empirique) = 2.3436

* Le pas optimal de la densité a noyau: H0 = 0.1472
* Le pas utilise : H0 = H0*coef (optimal) = 0.3385
* La somme des Tnoy[i] :      some = 1.0098

* la loi théorique est la loi Binomiale(19, 0.28)
    
```

xi	ni	pi	npi	khi2i
0	86	0.0020	20.4689	209.7976
1	300	0.0147	150.3805	148.8623
2	712	0.0513	523.3325	68.0169
3	1188	0.1124	1146.6977	1.4876
4	1715	0.1739	1773.5872	1.9353
5	1732	0.2017	2057.3936	51.4637
6	1516	0.1820	1856.2555	62.3695
7	1223	0.1307	1332.9893	9.0756
8	805	0.0758	773.1460	1.3124
9	449	0.0358	365.3888	19.1326
10	270	0.0139	141.2859	117.2609
11	116	0.0044	44.6984	113.7381
12	50	0.0011	11.5224	377.0651
13	25	0.0002	2.3991	--
14	5	0.0000	0.3976	--
15	6	0.0000	0.0512	--
16	1	0.0000	0.0050	--
17	0	0.0000	0.0003	--
18	1	0.0000	0.0000	--
19	0	0.0000	0.0000	--
sum	10200	1		1181.5175

* la statistique du Khi-deux calculée = 1181.5175
 * le degré de liberté de khi-deux = 11

Decision 1 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (11) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi Binomiale (19 , 0.28) (cf. Figure 33).

TEST D' HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ----> 32 x 32
Nombre total d'itérations         ( Kmax)    ----> 10200

Type de mode d'implémentation du réseau :
mode parallèle      --> note par 'par'
mode block-sequentiel --> note par 'bseq'
Votre choix exact          ----> bseq

Entrez le nombre total d'observations (Np)      ----> 17
Entrez le coefficient de correction du pas H0(coef)----> 2.30

Entrez le type de densité théorique :
Densité Normale(m, sigma)  --> Entrez 'n'
Densité Gamma(a,b)        --> Entrez 'g'
Densité Khi-deux a n d*1   --> Entrez 'k'
Densité Weibull(m,a,b)    --> Entrez 'w'
Densité de Poisson        --> Entrez 'p'
Densité Binomiale         --> Entrez 'b'
Votre choix                ----> b

Voulez-vous ouvrir le fichier densité a noyau(o/n) ----> n
Voulez-vous ouvrir le fichier densité théorique(o/n) ----> o
Nom du fichier lecture      ----> 32skhi2.dat
Nom du fichier écriture pour densité théorique ----> 32sbinoth.dat

* moyenne arithmétique      = 5.3223
* moyenne géométrique      = 4.7496
* SQRT(variance empirique) = 2.3534

* Le pas optimal de la densité a noyau: H0 = 0.1476
* Le pas utilisé : H0 = H0*coef (optimal) = 0.3394
* la somme des Tnoy[i] :      some = 1.0072

* La loi théorique est la loi Binomiale(16, 0.33)
    
```

xi	ni	pi	npi	khi2i
0	94	0.0015	15.7889	387.4209
1	280	0.0123	125.9184	188.5437
2	706	0.0461	470.7244	117.5945
3	1241	0.1073	1094.9382	19.4842
4	1616	0.1739	1773.7365	14.0273
5	1746	0.2080	2121.8588	66.5783
6	1529	0.1901	1938.9831	86.6878
7	1202	0.1354	1380.6771	23.1231
8	817	0.0759	774.2135	2.3646
9	495	0.0336	343.0239	67.3328
10	256	0.0117	119.6847	155.2569
11	118	0.0032	32.5397	224.4481
12	68	0.0007	6.7580	1071.6968
13	24	0.0001	1.0365	--
14	7	0.0000	0.1107	--
15	1	0.0000	0.0074	--
16	0	0.0000	0.0002	--
som	10200	1		2424.5590

```

* la statistique du Khi-deux calculée = 2424.5590
* le degré de liberté de khi-deux = 11
    
```

Decision 2 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (11) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi Binomiale (16 , 0.33) (cf. Figure 34).

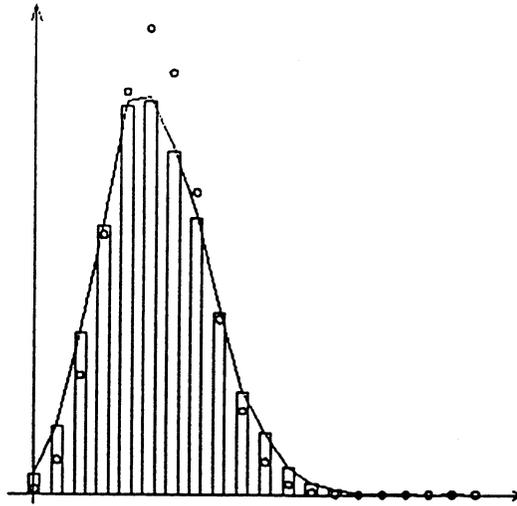


Figure 33 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi Binomiale ($B[16, 0.33]$) en symboles ronds dont le paramètre est estimé à partir de l'échantillon.

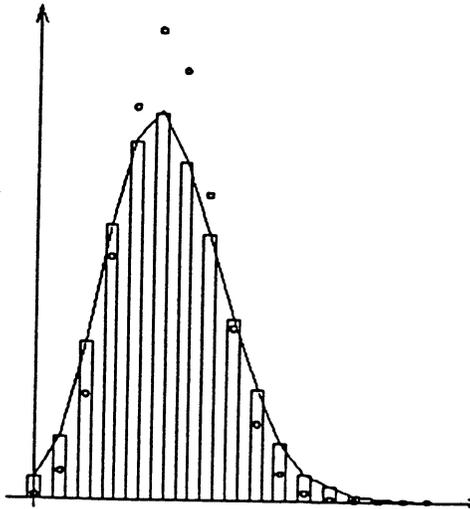


Figure 34 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi Binomiale ($B[16, 0.33]$) en symboles ronds dont le paramètre est estimé à partir de l'échantillon.

*** Test contre une loi de Gauss :**

```

TEST D'HYPOTHESE DU KHI-DEUX
-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations         ( Kmax)   ---> 10200
Type de mode d'implémentation du réseau :
  mode parallèle      --> note par 'par'
  mode block-séquentiel --> note par 'bseq'
                                Votre choix exact      ---> par
Entrez le nombre total d'observations (Np)      ---> 20
Entrez le coefficient de correction du pas H0(coef) ---> 2.30
Entrez le type de densité théorique :
  Densité Normale(m, sigma)  --> Entrez 'n'
  Densité Gamma(a,b)        --> Entrez 'g'
  Densité Khi-deux a n d'1   --> Entrez 'k'
  Densité Weibull(m,a,b)    --> Entrez 'w'
  Densité de Poisson        --> Entrez 'p'
  Densité Binomiale        --> Entrez 'b'
                                Votre choix            ---> n
Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> n
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture          ---> 32pkhi2.dat
Nom du fichier écriture pour densité théorique ---> 32pgauth.dat

```

```

* moyenne arithmétique = 5.2981
* moyenne géométrique = 4.7271
* SQRT(variance empirique) = 2.3436

```

```

* Test de normalité :
Si X une v.a. suivant une loi normale alors :
- Le Skewness = E((X-EX)**3)/sigma**3 = 0 (symétrie),
- Le Kurtosis = E((X-EX)**4)/sigma**4-3 = 0 (forme gauss),

```

loi gauss	Skewness	Kurtosis
théorique	0	0
calculé	0.403	0.233

* la loi théorique est la loi Gauss(5.30, 2.34)

xi	ni	pi	npi	khi2i
0	86	0.0136	139.0411	20.2340
1	300	0.0322	328.7795	2.5192
2	712	0.0637	649.7488	5.9642
3	1188	0.1052	1073.2049	12.2790
4	1715	0.1453	1481.5951	36.7697
5	1732	0.1676	1709.5698	0.2943
6	1516	0.1616	1648.8009	10.6963
7	1223	0.1303	1329.1195	8.4728
8	805	0.0878	895.5126	9.1484
9	449	0.0494	504.2953	6.0631
10	270	0.0233	237.3491	4.4916
11	116	0.0092	93.3605	5.4900
12	50	0.0030	30.6902	12.1493
13	25	0.0008	8.4312	68.4292
14	5	0.0002	1.9356	--
15	6	0.0000	0.3713	--
16	1	0.0000	0.0595	--
17	0	0.0000	0.0080	--
18	1	0.0000	0.0009	--
19	0	0.0000	0.0000	--
som	10200	1		203.0012

```

* la statistique du Khi-deux calculée = 203.0012
* le degré de liberté de khi-deux = 11

```

Decision 1 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (11) > A \alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A \alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Gauss (5.30 , 2.34) (cf. Figure 35).

TEST D'HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ----> 32 x 32
Nombre total d'itérations        ( Kmax)    ----> 10200
Type de mode d'implémentation du reseau :
  mode parallele                  --> note par 'par'
  mode block-sequentiel          --> note par 'bseq'
                                Votre choix exact ----> bseq
Entrez le nombre total d'observations (Np) ----> 17
Entrez le coefficient de correction du pas H0(coef)----> 2.30
Entrez le type de densite theorique :
  Densite Normale(m,sigma)       --> Entrez 'n'
  Densite Gamma(a,b)             --> Entrez 'g'
  Densite Khi-deux a n d°1       --> Entrez 'k'
  Densite Weibull(m,a,b)         --> Entrez 'w'
  Densite de Poisson             --> Entrez 'p'
  Densite Binomiale              --> Entrez 'b'
                                Votre choix ----> n
Voulez-vous ouvrir le fichier densite a noyau(o/n) ----> n
Voulez-vous ouvrir le fichier densite theoriq(o/n) ----> o
Nom du fichier lecture           ----> 32skhi2.dat
Nom du fichier ecriture pour densite theorique ----> 32sgausth.dat
  
```

```

* moyenne arithmétique = 5.3223
* moyenne géométrique = 4.7496
* SQRT(variance empirique) = 2.3534
  
```

```

* Test de normalite :
  Si X une v.a. suivant une loi normale alors :
  - Le Skewness=E((X-EX)**3)/sigma**3 = 0 (symétrie),
  - Le Kurtosis=E((X-EX)**4)/sigma**4-3 = 0 (forme gauss),
  
```

loi gauss	Skewness	Kurtosis
theorique	0	0
calculé	0.367	0.052

* la loi théorique est la loi Gauss(5.32, 2.35)

xi	ni	pi	npi	khi2i
0	94	0.0135	138.1793	14.1252
1	280	0.0319	325.8333	6.4471
2	706	0.0630	643.0842	6.1553
3	1241	0.1042	1062.3729	30.0343
4	1616	0.1440	1469.0480	14.6999
5	1746	0.1667	1700.3777	1.2241
6	1529	0.1615	1647.4750	8.5199
7	1202	0.1310	1336.1244	13.4638
8	817	0.0889	907.0411	8.9383
9	495	0.0505	515.4108	0.8083
10	256	0.0240	245.1380	0.4813
11	118	0.0096	97.5849	4.2709
12	68	0.0032	32.5131	38.7328
13	24	0.0009	9.0662	35.4630
14	7	0.0002	2.1158	--
15	1	0.0000	0.4132	--
16	0	0.0000	0.0675	--
som	10200	1		183.3643

```

* la statistique du Khi-deux calculee = 183.3643
* le degré de liberté de khi-deux = 11
  
```

Decision 2 :

$\forall \alpha > 0.001$ (le risque de 1ère espèce) tel que $P [\chi^2 (11) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Gauss (5.32 , 2.35) (cf. Figure 36).

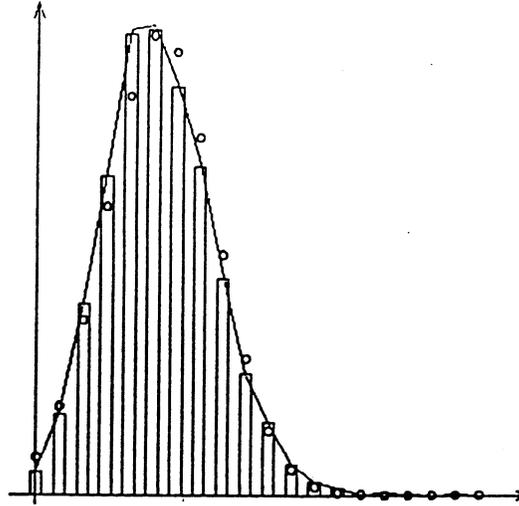


Figure 35 : Réseau de neurones de taille $N=32 \times 32$ en mode d'itération parallèle. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Gauss ($N [5.30, 2.34]$) en symboles ronds dont les paramètres sont estimés à partir de l'échantillon.

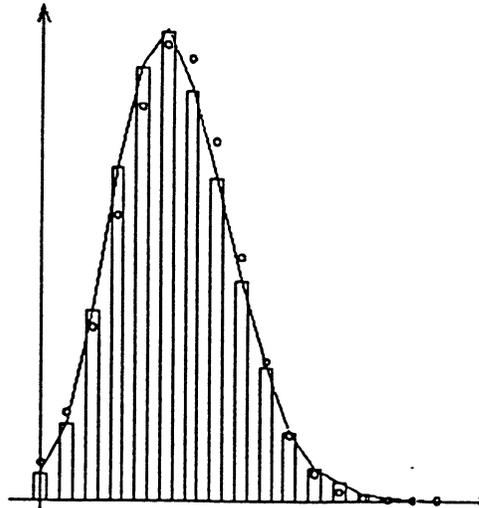


Figure 36 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Gauss ($N [5.32, 2.35]$) en symboles ronds dont les paramètres sont estimés à partir de l'échantillon.

*** Test contre une loi Gamma :**

```

TEST D'HYPOTHESE DU KHI-DEUX
-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations         ( Kmax)    ---> 10200
Type de mode d'implémentation du réseau :
  mode parallèle      --> note par 'par'
  mode block-sequentiel --> note par 'bseq'
                                Votre choix exact      ---> par
Entrez le nombre total d'observations (Np)      ---> 20
Entrez le coefficient de correction du pas H0(coef)---> 2.32
Entrez le type de densité théorique :
  Densité Normale(m, sigma)  --> Entrez 'n'
  Densité Gamma(a,b)         --> Entrez 'g'
  Densité Khi-deux a n d°1   --> Entrez 'k'
  Densité Weibull(m,a,b)     --> Entrez 'w'
  Densité de Poisson         --> Entrez 'p'
  Densité Binomiale          --> Entrez 'b'
                                Votre choix            ---> g
Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> o
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture      ---> 32pkhi2.dat
Nom du fichier écriture pour densité a noyau      ---> 32pgamano.dat
Nom du fichier écriture pour densité théorique    ---> 32pgamath.dat

* moyenne arithmétique      = 5.2981
* moyenne géométrique      = 4.7271
* SQRT(variance empirique) = 2.3436

```

* Estimations des paramètres d'une loi Gamma(a,b):

loi gamma	a	b
Moment	5.111	1.037
Max-Vrais	4.543	1.166
M.V (Newton)	4.544	1.166

* la loi théorique est la loi Gamma(4.54, 1.17)

xi	ni	pi	npi	khi2i
0	86	0.0008	8.3675	720.2658
1	300	0.0199	202.8122	46.5725
2	712	0.0855	872.1081	29.3938
3	1188	0.1511	1540.8547	80.8035
4	1715	0.1781	1816.6295	5.6856
5	1732	0.1673	1706.3129	0.3867
6	1516	0.1359	1386.3426	12.1262
7	1223	0.0999	1018.7524	40.9492
8	805	0.0682	695.5007	17.2395
9	449	0.0440	448.8445	0.0001
10	270	0.0272	277.0897	0.1814
11	116	0.0162	165.0263	14.5648
12	50	0.0094	95.4143	21.6158
13	25	0.0053	53.8106	15.4254
14	5	0.0029	29.7113	20.5527
15	6	0.0016	16.1082	6.3431
16	1	0.0008	8.5955	6.7118
17	0	0.0004	4.5231	6.2061
18	1	0.0002	2.3508	--
19	0	0.0001	1.2085	--
scm	10200	1		1045.0239

* la statistique du Khi-deux calculée = 1045.0239
 * le degré de liberté de khi-deux = 15

Decision 1 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (15) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi Gamma (4.54 , 1.17) (cf. Figure 37).

* la loi théorique est la loi Gamma(4.73, 1.13)

xi	ni	pi	npi	khi2i
1	300	0.0175	176.5837	86.2570
2	712	0.0815	824.7340	15.4098
3	1188	0.1508	1525.2833	74.5829
4	1715	0.1824	1845.2948	9.2000
5	1732	0.1738	1757.8160	0.3791
6	1516	0.1421	1437.4275	4.2949
7	1223	0.1046	1057.4689	25.9114
8	805	0.0712	719.9014	10.0594
9	449	0.0457	461.8908	0.3598
10	270	0.0280	282.8119	0.5804
11	116	0.0165	166.7349	15.4379
12	50	0.0094	95.2777	21.5168
13	25	0.0052	53.0357	14.8202
14	5	0.0029	28.8702	19.7361
15	6	0.0015	15.4161	5.7514
16	1	0.0008	8.0952	6.2188
17	0	0.0004	4.1888	5.5433
18	1	0.0002	2.1394	--
19	0	0.0001	1.0801	--
som	10114	1		316.0591

* la statistique du Khi-deux calculée = 316.0591
 * le degré de liberté de khi-deux = 14

Decision 2 :

$\forall \alpha > 0$ (le risque de 1^{ère} espèce) tel que $P [\chi^2(14) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T < A_\alpha$. Donc, on rejette toujours l'hypothèse que la loi soit une loi Gamma (4.73 , 1.13) (cf. Figure 37).

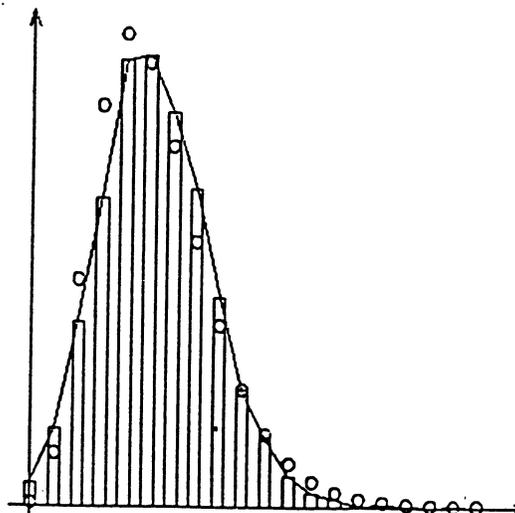


Figure 37 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi Gamma ($\Gamma [4.54 , 1.16]$) en symboles ronds dont les paramètres sont estimés à partir de l'échantillon. Si on supprime la première observation "0-uplet", on rejette toujours l'hypothèse d'une loi gamma. Par contre, on remarque que la statistique du khi-deux a diminué plus de 70% par rapport à celle de l'échantillon non tronqué (voir tableau juste au-dessus).

TEST D'HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille          ( taille)  ---> 32 x 32
Nombre total d'itérations              ( Kmax)    ---> 10200
Type de mode d'implémentation du réseau :
  mode parallèle                       --> note par 'par'
  mode block-séquentiel --> note par 'bseq'
  Votre choix exact                     ---> bseq
Entrez le nombre total d'observations (Np) ---> 17
Entrez le coefficient de correction du pas H0(coef) ---> 2.32

Entrez le type de densité théorique :
  Densité Normale(m,sigma)             --> Entrez 'n'
  Densité Gamma(a,b)                   --> Entrez 'g'
  Densité Khi-deux a n d°1             --> Entrez 'k'
  Densité Weibull(m,a,b)               --> Entrez 'w'
  Densité de Poisson                   --> Entrez 'p'
  Densité Binomiale                    --> Entrez 'b'
  Votre choix                           ---> g
Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> o
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture                  ---> 32skhi2.dat
Nom du fichier écriture pour densité a noyau ---> 32sgamano.dat
Nom du fichier écriture pour densité théorique ---> 32sgamath.dat
  
```

```

* moyenne arithmétique = 5.3223
* moyenne géométrique = 4.7496
* SQRT(variance empirique) = 2.3534
  
```

* Estimations des paramètres d'une loi Gamma(a,b):

loi gamma	a	b
Moment	5.114	1.041
Max-Vrais	4.552	1.169
M.V (Newton)	4.553	1.169

* la loi théorique est la loi Gamma(4.55, 1.17)

xi	ni	pi	npi	khi2i
0	94	0.0008	8.0963	911.4565
1	280	0.0195	198.6452	33.3187
2	706	0.0843	860.1076	27.6118
3	1241	0.1498	1527.9716	53.8967
4	1616	0.1774	1809.8128	20.7554
5	1746	0.1673	1706.9598	0.8929
6	1529	0.1365	1392.1618	13.4501
7	1202	0.1007	1026.6953	29.9327
8	817	0.0690	703.3146	18.3764
9	495	0.0446	455.3754	3.4479
10	256	0.0276	282.0136	2.3996
11	118	0.0165	168.4774	15.1235
12	68	0.0096	97.7036	9.0304
13	24	0.0054	55.2646	17.6872
14	7	0.0030	30.6028	18.2039
15	1	0.0016	16.6390	14.6991
16	0	0.0009	8.9038	8.9038
som	10200	1		1199.1867

```

* la statistique du Khi-deux calculée = 1199.1867
* le degré de liberté de khi-deux = 14
  
```

Decision 3 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2(14) > A \alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A \alpha$. Donc, on rejette l'hypothèse que la loi soit une loi Gamma (4.55 , 1.17) (cf. Figure 38).

* la loi théorique est la loi Gamma(4.76, 1.13)

xi	ni	pi	npi	khi2i
1	280	0.0169	170.4538	70.4025
2	706	0.0801	809.1065	13.1391
3	1241	0.1497	1512.4009	48.7030
4	1616	0.1824	1843.6572	28.1114
5	1746	0.1748	1766.4229	0.2361
6	1529	0.1436	1451.0971	4.1823
7	1202	0.1060	1071.5301	15.8861
8	817	0.0724	731.7588	9.9296
9	495	0.0466	470.7459	1.2496
10	256	0.0286	288.8913	3.7448
11	118	0.0169	170.6557	16.2469
12	68	0.0097	97.6863	9.0215
13	24	0.0054	54.4587	17.0355
14	7	0.0029	29.6843	17.3350
15	1	0.0016	15.8695	13.9325
16	0	0.0008	8.3420	8.3420
som	10106	1		277.4979

* la statistique du Khi-deux calculée = 277.4979
 * le degré de liberté de khi-deux = 13

Decision 4 :

$\forall \alpha > 0$ (le risque de 1ère espèce) tel que $P [\chi^2(12) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette toujours l'hypothèse que la loi soit une loi Gamma (4.76 , 1.13) (cf. Figure 38).

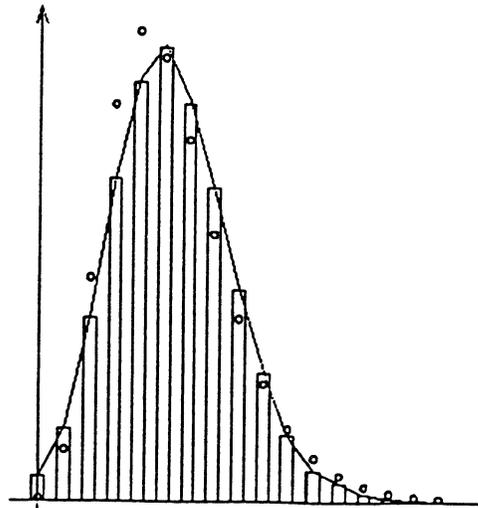


Figure 38 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi Gamma ($\Gamma [4.55, 1.17]$) en symboles ronds dont les paramètres sont estimés à partir de l'échantillon. Si on supprime la première observation "0-uplet", on rejette toujours l'hypothèse d'une loi gamma. Par contre, on remarque que la statistique du khi-deux a diminué plus de 75% par rapport à celle de l'échantillon non tronqué (voir tableau juste au-dessus).

*** Test contre une loi de Khi-deux :**

```

TEST D' HYPOTHESE DU KHI-DEUX
-----
Réseau de neurones de taille          ( taille)  ---> 32 x 32
Nombre total d'itérations              ( Kmax)    ---> 10200
Type de mode d'implémentation du réseau :
  mode parallèle                       --> note par 'par'
  mode block-sequentiel                 --> note par 'bseq'
  Votre choix exact                     ---> par

Entrez le nombre total d'observations (Np)  ---> 20
Entrez le coefficient de correction du pas H0(coef) ---> 2.30

Entrez le type de densité théorique :
  Densité Normale(m,sigma)             --> Entrez 'n'
  Densité Gamma(a,b)                   --> Entrez 'g'
  Densité Khi-deux a n d°1              --> Entrez 'k'
  Densité Weibull(m,a,b)                --> Entrez 'w'
  Densité de Poisson                    --> Entrez 'p'
  Densité Binomiale                     --> Entrez 'b'
  Votre choix                           ---> k
Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> o
Voulez-vous ouvrir le fichier densité theoriq(o/n) ---> o
Nom du fichier lecture                  ---> 32pkhi2.dat
Nom du fichier écriture pour densité a noyau ---> 32pkhi2no.dat
Nom du fichier écriture pour densité théorique ---> 32pkhi2th.dat

* moyenne arithmétique = 5.2981
* moyenne géométrique = 4.7271
* SQRT(variance empirique) = 2.3436

* la loi théorique est la loi Khi2( 5 )

```

xi	ni	pi	npi	khi2i
0	86	0.0191	195.1918	61.0828
1	300	0.0791	806.3426	317.9577
2	712	0.1366	1393.0588	332.9659
3	1188	0.1531	1561.6937	89.4202
4	1715	0.1435	1463.7418	43.1297
5	1732	0.1219	1243.7629	191.6567
6	1516	0.0974	993.3519	274.9892
7	1223	0.0745	760.1923	281.7590
8	805	0.0553	563.8772	103.1079
9	449	0.0400	408.4124	4.0336
10	270	0.0285	290.3072	1.4205
11	116	0.0199	203.2464	37.4517
12	50	0.0138	140.5226	58.3133
13	25	0.0094	96.1395	52.6405
14	5	0.0064	65.1883	55.5718
15	6	0.0043	43.8619	32.6827
16	1	0.0029	29.3150	27.3491
17	0	0.0019	19.4773	19.4773
18	1	0.0013	12.8736	10.9513
19	0	0.0008	8.4693	8.4693
som	10200	1		2004.4302

* la statistique du Khi-deux calculée = 2004.4302
 * le degré de liberté de khi-deux = 19

Decision 1 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (19) > A \alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A \alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Khi-deux (5) (cf. Figure 39).

TEST D' HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations         ( Kmax)    ---> 10200

Type de mode d'implémentation du réseau :
mode parallèle      --> note par 'par'
mode block-sequentiel --> note par 'bseq'
Votre choix exact          ---> bseq

Entrez le nombre total d'observations (Np)      ---> 17
Entrez le coefficient de correction du pas H0(coef) ---> 2.30

Entrez le type de densité théorique :
Densité Normale(m, sigma)  --> Entrez 'n'
Densité Gamma(a,b)        --> Entrez 'g'
Densité Khi-deux a n d'1  --> Entrez 'k'
Densité Weibull(m,a,b)    --> Entrez 'w'
Densité de Poisson        --> Entrez 'p'
Densité Binomiale         --> Entrez 'b'
Votre choix                ---> k

Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> o
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture      ---> 32skhi2.dat
Nom du fichier écriture pour densité a noyau ---> 32skhi2no.dat
Nom du fichier écriture pour densité théorique ---> 32skhi2th.dat

* moyenne arithmétique      = 5.3223
* moyenne géométrique      = 4.7496
* SQRT(variance empirique) = 2.3534
    
```

* la loi théorique est la loi Khi2(5)

xi	ni	pi	npi	khi2i
0	94	0.0191	195.1918	52.4601
1	280	0.0791	806.3426	343.5717
2	706	0.1366	1393.0588	338.8585
3	1241	0.1531	1561.6937	65.8544
4	1616	0.1435	1463.7418	15.8379
5	1746	0.1219	1243.7629	202.8057
6	1529	0.0974	993.3519	288.8391
7	1202	0.0745	760.1923	256.7694
8	817	0.0553	563.8772	113.6261
9	495	0.0400	408.4124	18.3575
10	256	0.0285	290.3072	4.0543
11	118	0.0199	203.2464	35.7544
12	68	0.0138	140.5226	37.4283
13	24	0.0094	96.1395	54.1308
14	7	0.0064	65.1883	51.9400
15	1	0.0043	43.8619	41.8847
16	0	0.0029	29.3150	29.3150
som	10200	1		1951.4878

* la statistique du Khi-deux calculée = 1951.4878
 * le degré de liberté de khi-deux = 16

Decision 2 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2 (16) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Khi-deux (5) (cf. Figure 40).

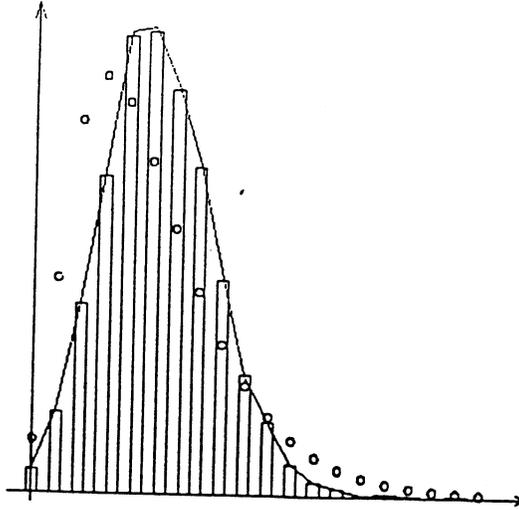


Figure 39 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Khi-deux ($\chi^2[5]$) en symboles ronds.

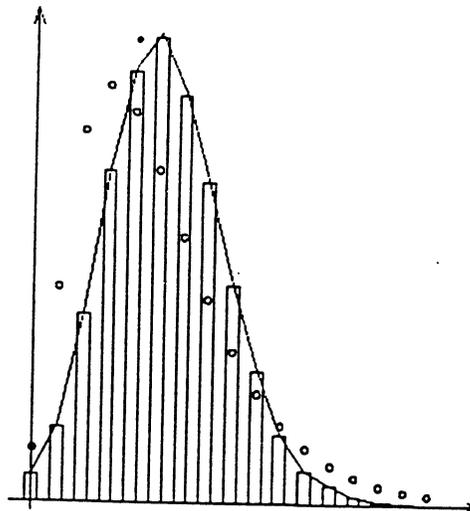


Figure 40 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Khi-deux ($\chi^2[5]$) en symboles ronds.

** Test contre une loi de Weibull :*

TEST D'HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations         ( Kmax)   ---> 10200
Type de mode d'implémentation du réseau :
  mode parallèle                   --> note par 'par'
  mode block-sequentiel            --> note par 'bseq'
  Votre choix exact                 ---> par
Entrez le nombre total d'observations (Np)  ---> 20
Entrez le coefficient de correction du pas H0(coef)---> 2.32
Entrez le type de densité théorique :
  Densité Normale(m,sigma)         --> Entrez 'n'
  Densité Gamma(a,b)               --> Entrez 'g'
  Densité Khi-deux a n d'1         --> Entrez 'k'
  Densité Weibull(m,a,b)           --> Entrez 'w'
  Densité de Poisson                --> Entrez 'p'
  Densité Binomiale                 --> Entrez 'b'
  Votre choix                       ---> w
Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> o
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture              ---> 32pkhi2.dat
Nom du fichier écriture pour densité a noyau ---> 32pweibno.dat
Nom du fichier écriture pour densité théorique ---> 32pweibth.dat

* moyenne arithmétique      = 5.2981
* moyenne géométrique      = 4.7271
* SORT(variance empirique) = 2.3436
    
```

* Estimations des paramètres d'une loi weibull(m,a,c):

loi weibull	m	a	c
Graphique	0.000	11.443	1.849
Max-Vrais	0.000	5.984	2.429

* La loi théorique est la loi Weibull(0.00, 5.98, 2.43)

xi	ni	pi	npi	khi2i
0	86	0.0024	24.5195	154.1565
1	300	0.0317	323.3698	1.6889
2	712	0.0790	805.6233	10.8802
3	1188	0.1249	1273.6056	5.7540
4	1715	0.1557	1588.4383	10.0840
5	1732	0.1635	1667.8557	2.4669
6	1516	0.1483	1512.2216	0.0094
7	1223	0.1173	1196.7529	0.5756
8	805	0.0814	830.1441	0.7616
9	449	0.0495	505.2054	6.2530
10	270	0.0264	269.5157	0.0009
11	116	0.0123	125.8056	0.7643
12	50	0.0050	51.2575	0.0308
13	25	0.0018	18.1780	2.5602
14	5	0.0005	5.5946	7.5068
15	6	0.0001	1.4896	--
16	1	0.0000	0.3420	--
17	0	0.0000	0.0675	--
18	1	0.0000	0.0114	--
19	0	0.0000	0.0017	--
scm	10200	1		200.0062

* La statistique du Khi-deux calculée = 200.0062
 * le degré de liberté de khi-deux = 11

Decision 1 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2(12) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Weibull [0.0 , 5.98 , 2.43] (cf. Figure 41).

* la loi théorique est la loi Weibull(0.00, 6.02, 2.48)

xi	ni	pi	npi	khi2i
1	300	0.0294	297.2089	0.0262
2	712	0.0756	765.1066	3.6862
3	1188	0.1223	1237.1173	1.9501
4	1715	0.1552	1569.7598	13.4382
5	1732	0.1651	1669.9637	2.3045
6	1516	0.1511	1527.7234	0.0900
7	1223	0.1201	1214.2739	0.0627
8	805	0.0832	841.5420	1.5868
9	449	0.0503	508.6453	6.9942
10	270	0.0265	267.6948	0.0199
11	116	0.0121	122.3518	0.3298
12	50	0.0048	48.4092	0.0523
13	25	0.0016	16.5213	4.3513
14	5	0.0005	4.8453	6.8767
15	6	0.0001	1.2164	--
16	1	0.0000	0.2604	--
17	0	0.0000	0.0473	--
18	1	0.0000	0.0073	--
19	0	0.0000	0.0009	--
som	10114	1		41.7687

* la statistique du Khi-deux calculée = 41.7687
 * le degré de liberté de khi-deux = 11

Decision 2 :

$\forall \alpha > 0$ (le risque de 1^{ère} espèce) tel que $P [\chi^2(12) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Weibull [0.0 , 6.02 , 2.48] (cf. Figure 41).

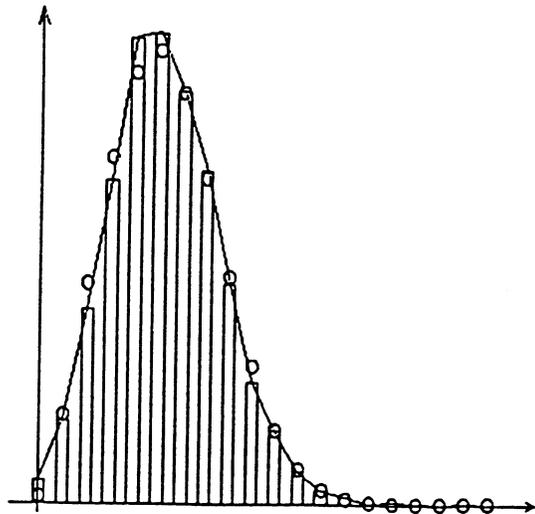


Figure 41 : Réseau de neurones de taille $N=32 \times 32$ en mode d'itération parallèle. Représentation simultanée des graphiques suivants: l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Weibull ($W [0.0, 5.98, 2.43]$) en symboles ronds dont les paramètres sont estimés à partir de l'échantillon. Même quand on a supprimé la première observation "0-uplet", voir tableau juste au-dessus, on a toujours rejeté l'hypothèse d'une loi de weibull. Par contre, on remarque que la statistique du khi-deux a diminué plus de 80% par rapport à celle de l'échantillon non tronqué (voir tableau juste au dessus).

TEST D'HYPOTHESE DU KHI-DEUX

```

-----
Réseau de neurones de taille      ( taille)  ---> 32 x 32
Nombre total d'itérations        ( Kmax)   ---> 10200

Type de mode d'implémentation du réseau :
mode parallèle      --> note par 'par'
mode block-séquentiel --> note par 'bseq'
Votre choix exact      ---> bseq

Entrez le nombre total d'observations (Np)      ---> 17
Entrez le coefficient de correction du pas H0(coef) ---> 2.32
Entrez le type de densité théorique :
Densité Normale(m, sigma)  --> Entrez 'n'
Densité Gamma(a,b)        --> Entrez 'g'
Densité Khi-deux a n d'1  --> Entrez 'k'
Densité Weibull(m,a,b)    --> Entrez 'w'
Densité de Poisson        --> Entrez 'p'
Densité Binomiale         --> Entrez 'b'
Votre choix                ---> w
Voulez-vous ouvrir le fichier densité a noyau(o/n) ---> n
Voulez-vous ouvrir le fichier densité théorique(o/n) ---> o
Nom du fichier lecture      ---> 32skhi2.dat
Nom du fichier écriture pour densité théorique ---> 32sweibth.dat

* moyenne arithmétique      = 5.3223
* moyenne géométrique      = 4.7496
* SQRT(variance empirique) = 2.3534
    
```

* Estimations des paramètres d'une loi weibull(m, a, c):

loi weibull	m	a	c
Graphique	0.000	9.770	1.906
Max-Vrais	0.000	6.013	2.436

* La loi théorique est la loi Weibull(0.00, 6.01, 2.44)

xi	ni	pi	npi	khi2i
0	94	0.0023	23.8064	206.9665
1	280	0.0311	316.7612	4.2663
2	706	0.0778	793.5970	9.6689
3	1241	0.1236	1260.3144	0.2960
4	1616	0.1548	1578.7352	0.8796
5	1746	0.1632	1665.0455	3.9360
6	1529	0.1487	1516.6233	0.1010
7	1202	0.1182	1205.9439	0.0129
8	817	0.0824	840.5951	0.6623
9	495	0.0504	514.0906	0.7089
10	256	0.0270	275.6095	1.3952
11	118	0.0127	129.2764	0.9836
12	68	0.0052	52.9207	4.2967
13	24	0.0018	18.8526	1.4054
14	7	0.0006	5.8267	0.0085
15	1	0.0002	1.5574	--
16	0	0.0000	0.3588	--
scm	10200	1		235.5879

* la statistique du Khi-deux calculée = 235.5879
 * le degré de liberté de khi-deux = 11

Decision 3 :

$\forall \alpha > 0.001$ (le risque de 1^{ère} espèce) tel que $P [\chi^2(11) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T > A_\alpha$. Donc, on rejette l'hypothèse que la loi soit une loi de Weibull [0.0, 6.01, 2.44] (cf. Figure 42).

* La loi théorique est la loi Weibull(0.00, 6.06, 2.49)

xi	ni	pi	npi	khi2i
1	280	0.0285	288.3402	0.2412
2	706	0.0741	749.2002	2.4910
3	1241	0.1207	1219.9956	0.3616
4	1616	0.1541	1557.8222	2.1727
5	1746	0.1650	1667.1339	3.7309
6	1529	0.1518	1533.7760	0.0149
7	1202	0.1213	1225.5759	0.4535
8	817	0.0845	853.5132	1.5620
9	495	0.0513	518.0883	1.0289
10	256	0.0271	273.6265	1.1355
11	118	0.0124	125.3897	0.4355
12	68	0.0049	49.6869	6.7497
13	24	0.0017	16.9618	2.9204
14	7	0.0005	4.9687	0.3578
15	1	0.0001	1.2439	--
16	0	0.0000	0.2651	--
som	10106	1		23.6556

* la statistique du Khi-deux calculée = 23.6556
 * le degré de liberté de khi-deux = 11

Decision 4 :

pour $\alpha = 1\%$ (le risque de 1^{ère} espèce) tel que $P[\chi^2(11) > A_\alpha] = \alpha$, la statistique du Khi-deux " T " est telle que : $T < A_\alpha$. Donc, on accepte l'hypothèse que la loi soit une loi de Weibull [0.0, 6.06, 2.49] (cf. Figure 42).

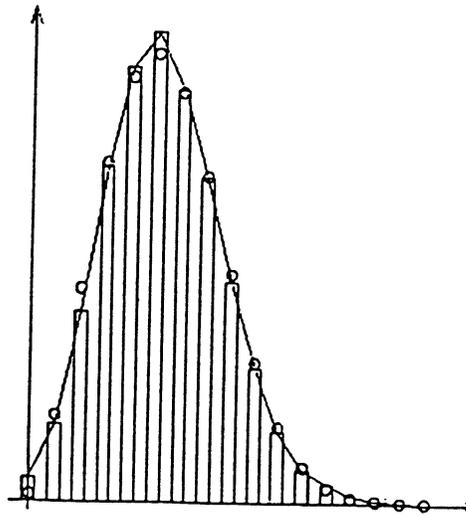


Figure 42 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Représentation simultanée des graphiques suivants : l'histogramme de l'échantillon (non tronqué), l'estimation à noyau, en ligne continue, et la densité théorique de la loi de Weibull ($W [0.0, 6.01, 2.44]$) en symboles ronds dont les paramètres sont estimés à partir de l'échantillon. Par contre dans le tableau juste au-dessus, on a supprimé la première observation "0-uplet", ce qui a induit une acceptation de l'hypothèse d'une loi de Weibull pour un risque de 1 %.

IV -3.2.2 Temps de convergence

* Calcul de la longueur du transitoire

On laisse évoluer des systèmes de réseaux de neurones de tailles différentes (cf. VI - 3.2.1 § 1°), tenant compte des aspects temporels, spatiaux et stochastiques de l'information neuronale ; plus précisément, ces systèmes évoluent selon un critère "Markovien-spatial", i.e.

$$Prob\{x_i(t) \in \{0, 1\} \mid X = (x_i(t-1))\} = Prob\{x_i(t) \in \{0, 1\} \mid X \cap N_i(t-1)\}, \text{ avec :}$$

Si l'on pose : $i = (i_1, i_2)$, sont les coordonnées du site "i" dans la boîte Ω du réseau telle que :

$$N_i = \{ j ; d[(i_1, i_2), (j_1, j_2)] \leq 2 \} \text{ où } d = |i_1 - j_1| + |i_2 - j_2| \text{ est appelée distance de Manhattan.}$$

Cette Probabilité de transition, locale (en chaque site), dépend de la connectique inter-neuronale des poids d'inter-connections et du seuil du neurone "i" lui-même. Ces systèmes convergent vers un état stable de Gibbs en réponse à un champ d'entrée stationnaire. L'état de Gibbs est obtenu après un régime transitoire dont on essaiera dans la suite de déterminer la fin.

Le problème majeur du processus de simulation est donc : comment choisir les règles (ou critères) d'arrêt déterminant la fin du transitoire du processus neuronal ? Dans notre cas, si le processus $x(t)$ est ergidique, alors le temps de convergence est déterminé dès l'installation de la mesure de Gibbs. Pour cela, on a choisi deux critères d'arrêts basés sur la stabilité des estimations des potentiels d'interaction, selon les situations des réseaux en question :

- Le premier cas consiste à tester la stabilité des configurations de singletons (ou des configurations *1-uplets*). La convergence du système est donc basée sur la convergence de la statistique " $I_{cv}(t, t')$ " (ou indice de convergence, ou encore écart moyen relatif) :

$$I_{cv}(t, t') = \frac{1}{N} \sum_{1\text{-uplets}} \frac{|J_U^*(A, t) - J_U^*(A, t-t')|}{|J_U^*(A, t)|}, \quad \text{[(B)-1]}$$

où $J_U^*(A, t) = \text{Log}(N_A) - \text{Log}(N_\emptyset)$ et où N_A, N_\emptyset sont respectivement les fréquences d'apparitions de "A" et du vide à l'instant "t", N étant le nombre de neurones dans le réseau.

Cet indice de convergence est appliqué seulement à des réseaux de tailles 8×8 et 16×16 , cela pour une simple raison : c'est que, dans les deux tailles, les configurations de singletons observées sont importantes (cf. IV -3.2.1 §1°).

En ce qui concerne l'évolution de "Icv" en fonction de "t'", selon les tailles des réseaux et les modes d'implémentations, voir les graphiques ci-dessous (cf. Figures 43, 44, 45, 46, 47 et 48).

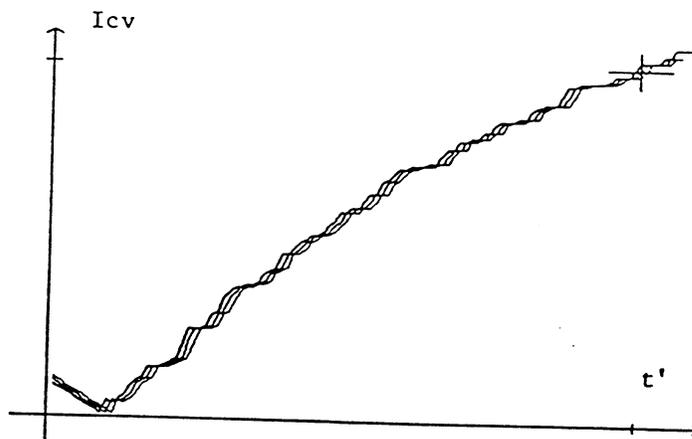


Figure 43 : Réseau de neurones de taille $N=8 \times 8$ en mode d'implémentation parallèle. Evolution de l'indice de convergence $Icv(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', Icv(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t = Tcv-1, Tcv$ et $Tcv+1$, où $Tcv = 2595$ itérations données par la relation ((B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t = Tcv-1$ ne vérifie pas la condition de la relation ((B)-3). Donc, la trajectoire associée à la valeur $t = Tcv$ est bien minimale au sens de la relation ((B)-3).

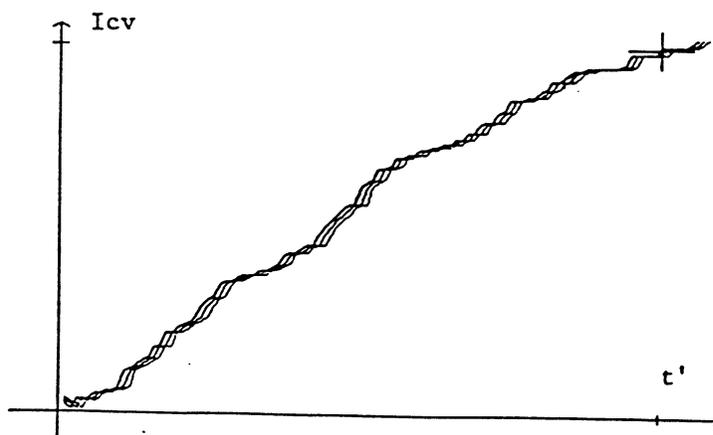


Figure 44 : Réseau de neurones de taille $N=8 \times 8$ en mode d'implémentation bloc-séquentiel. Evolution de l'indice de convergence $Icv(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', Icv(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=Tcv-1, Tcv$ et $Tcv+1$, où $Tcv=2599$ itérations données par la relation ((B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=Tcv-1$ ne vérifie pas la condition de la relation ((B)-3). Donc, la trajectoire associée à la valeur $t=Tcv$ est bien minimale au sens de la relation ((B)-3).

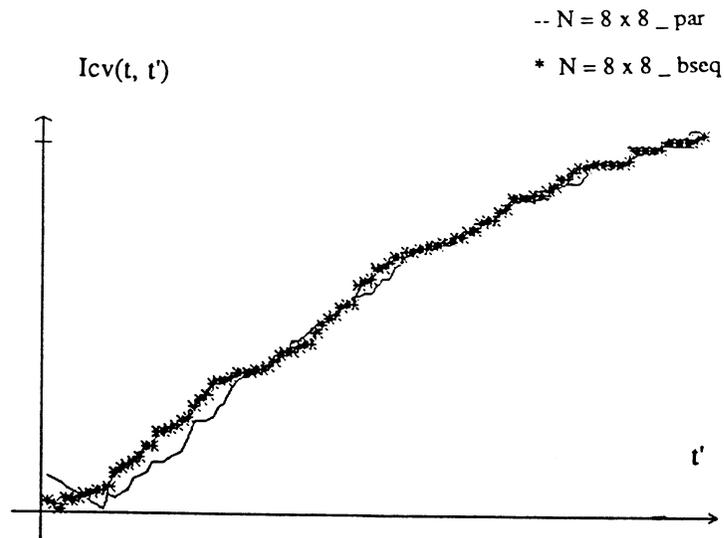


Figure 45 : Réseau de neurones de taille $N=8 \times 8$. Evolution de l'indice de convergence $Icv(t, t')$ en fonction de t' , avec $t=Tcv_s$ (correspond au réseau en mode d'implémentation bloc-séquentiel) et $t=Tcv_p$ (correspond au réseau en mode d'implémentation parallèle) fixés (cf. (B)-1).

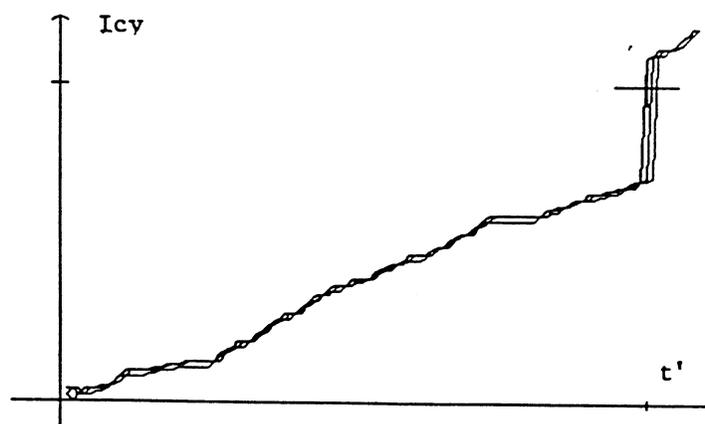


Figure 46 : Réseau de neurones de taille $N=16 \times 16$ en mode d'implémentation parallèle. Evolution de l'indice de convergence $Icy(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100$, $\alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', Icy(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=Tcv-1$, Tcv et $Tcv+1$, où $Tcv=3690$ itérations qui sont données par la relation (B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=Tcv-1$ ne vérifie pas la condition de la relation (B)-3). Donc, la trajectoire associée à la valeur $t=Tcv$ est bien minimale au sens de la relation (B)-3).

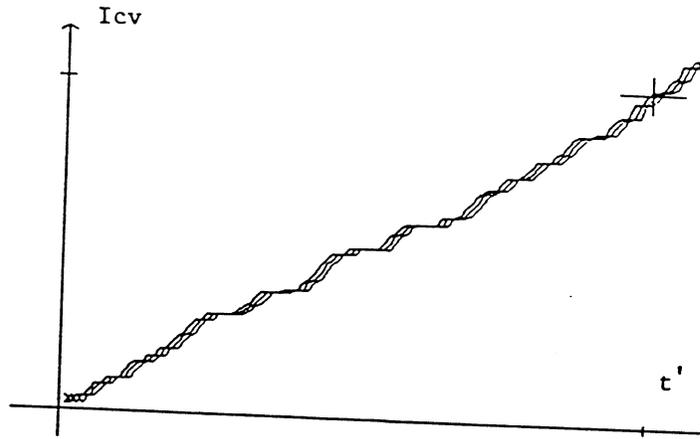


Figure 47 : Réseau de neurones de taille $N=16 \times 16$ en mode d'implémentation bloc-séquentiel. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', I_{cv}(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=T_{cv}-1, T_{cv}$ et $T_{cv}+1$, où $T_{cv}=2753$ itérations qui sont données par la relation (B)-3 (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=T_{cv}-1$ ne vérifie pas la condition de la relation (B)-3. Donc, la trajectoire associée à la valeur $t=T_{cv}$ est bien minimale au sens de la relation (B)-3).

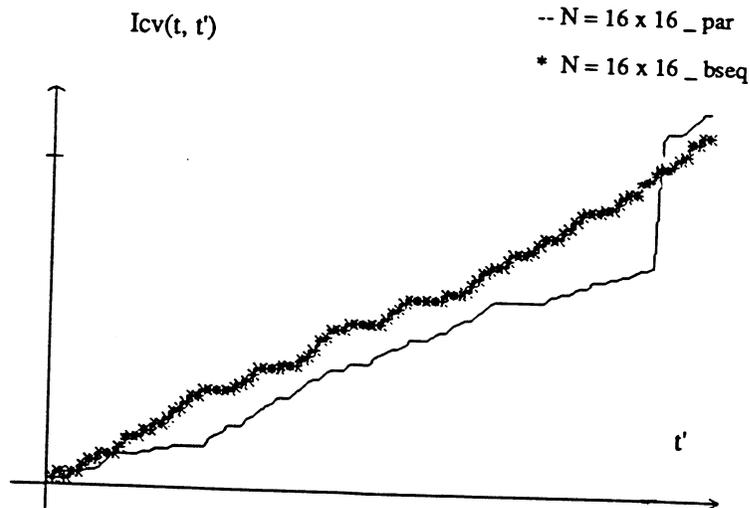


Figure 48 : Réseau de neurones de taille $N=16 \times 16$. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec $t=T_{cvs}$ (correspond au réseau en mode d'implémentation bloc-séquentiel) et $t=T_{cvp}$ (correspond au réseau en mode d'implémentation parallèle) fixés (cf. (B)-1).

En résumé, dans ce premier cas, l'indice de convergence croît plus vite initialement en mode séquentiel, mais les temps d'arrêt sont sensiblement les mêmes.

- Le second cas, concernant les autres tailles, consiste à tester la stabilité des potentiels des configurations les plus apparues dans les réseaux correspondants.

Exemple : dans le réseau de taille $N=32 \times 32$ neurones, les configurations les plus apparues sont les configurations 5-uplets, etc... (cf. § IV 3.2.1 § 1°). et l'indice de convergence est défini par :

$$I'_{cv}(t, t') = \sum_{A \in C(A)} \frac{|U^*(A, t) - U^*(A, t-t')|}{|U^*(A, t)|} \quad [(B)-2]$$

où $C(A)$ désignant l'ensemble des C_N^M configurations dont le cardinal M et le plus apparue, et $U^*(A, t) = \text{Log}(N_A) - \text{Log}(N_\emptyset)$ (cf. 1° cas).

En ce qui concerne l'évolution de " I'_{cv} " en fonction de " t' ", selon les tailles des réseaux et les modes d'implémentation, voir graphiques ci-dessous (cf. Figures 49, 50, 51, 52, 53, 54, 55, 56 et 57).

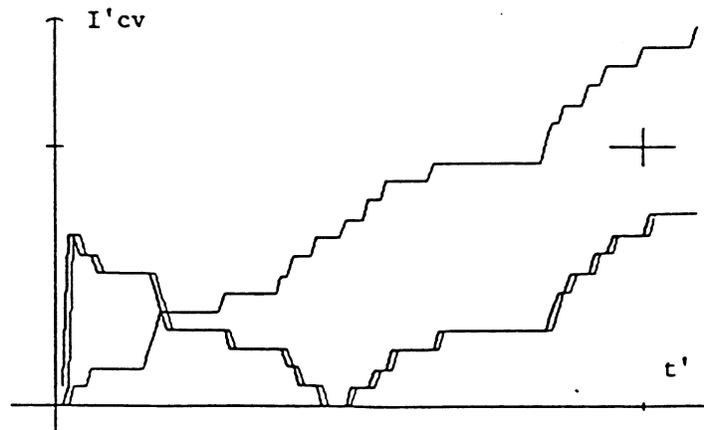


Figure 49 : Réseau de neurones de taille $N=32 \times 32$ en mode d'implémentation parallèle. Evolution de l'indice de convergence $I'_{cv}(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', I'_{cv}(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=T_{cv}-1, T_{cv}$ et

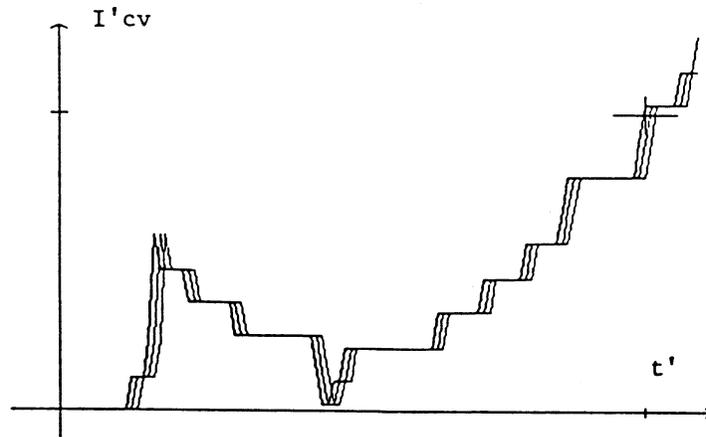


Figure 50 : Réseau de neurones de taille $N=32 \times 32$ en mode d'implémentation bloc-séquentiel. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', I_{cv}(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=T_{cv}-1, T_{cv}$ et $T_{cv}+1$, où $T_{cv}=2719$ itérations qui sont données par la relation (B)-3 (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=T_{cv}-1$ ne vérifie pas la condition de la relation (B)-3). Donc, la trajectoire associée à la valeur $t=T_{cv}$ est bien minimale au sens de la relation (B)-3).

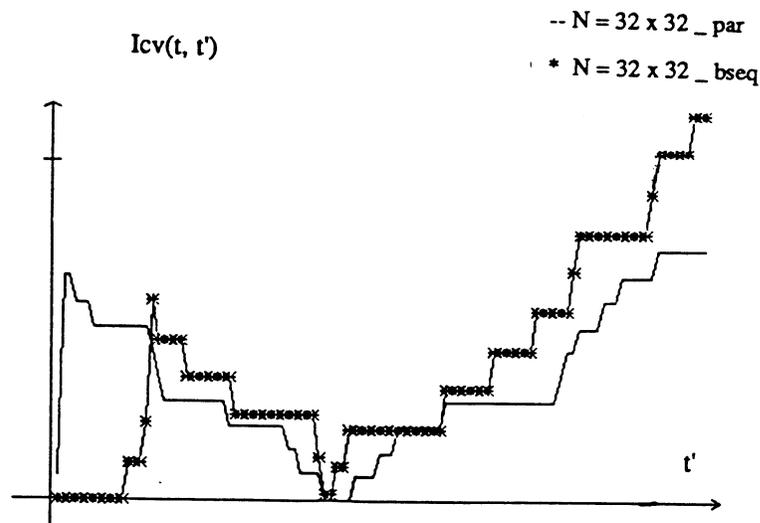


Figure 51 : Réseau de neurones de taille $N=32 \times 32$. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec $t=T_{cv}$ (correspond au réseau en mode d'implémentation bloc-séquentiel) et $t=T_{cvp}$ (correspond au réseau en mode d'implémentation parallèle) fixés (cf. (B)-1).

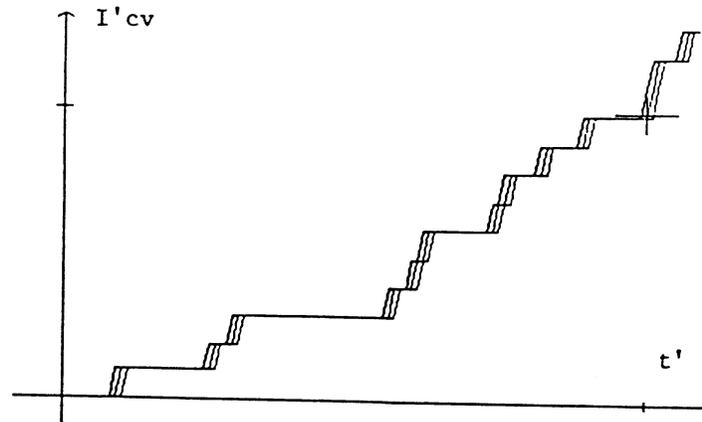


Figure 52 : Réseau de neurones de taille $N=64 \times 64$ en mode d'implémentation parallèle. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', I_{cv}(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=T_{cv}-1, T_{cv}$ et $T_{cv}+1$, où $T_{cv} = 4515$ itérations qui sont données par la relation (B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=T_{cv}-1$ ne vérifie pas la condition de la relation (B)-3). Donc, la trajectoire associée à la valeur $t=T_{cv}$ est bien minimale au sens de la relation (B)-3).

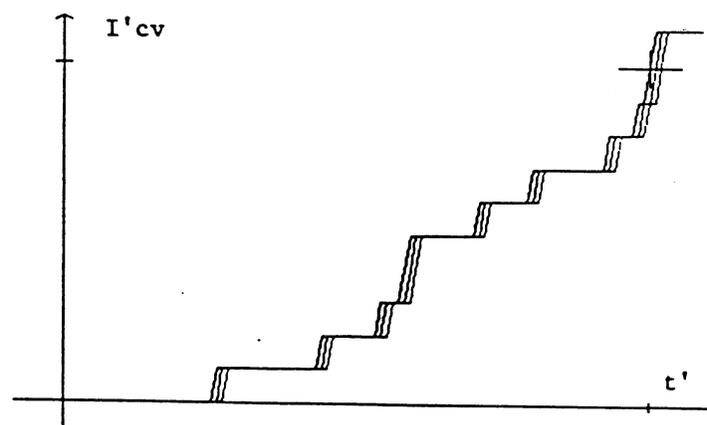


Figure 53 : Réseau de neurones de taille $N=64 \times 64$ en mode d'implémentation bloc-séquentiel. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$) (cf. (B)-3). Les trajectoires $(t', I_{cv}(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=T_{cv}-1, T_{cv}$ et $T_{cv}+1$, où $T_{cv}=4935$ itérations qui sont données par la relation (B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=T_{cv}-1$ ne vérifie pas la condition de la relation (B)-3). Donc, la trajectoire associée à la valeur $t=T_{cv}$ est bien minimale au sens de la relation (B)-3).

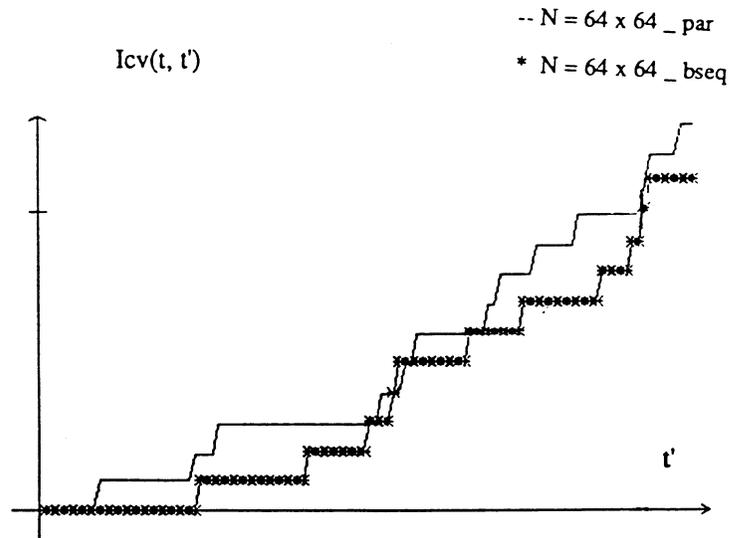


Figure 54 : Réseau de neurones de taille $N = 64 \times 64$. Evolution de l'indice de convergence $I'cv(t, t')$ en fonction de t' , avec $t = Tcv_s$ (correspond au réseau en mode d'implémentation bloc-séquentiel) et $t = Tcv_p$ (correspond au réseau en mode d'implémentation parallèle) fixés (cf. (B)-1).

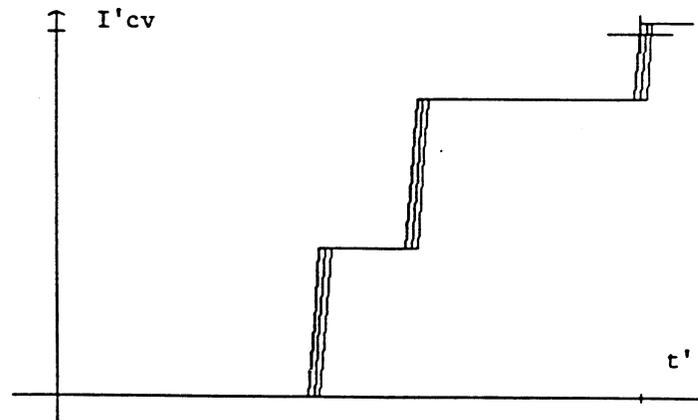


Figure 55 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'implémentation parallèle. Evolution de l'indice de convergence $I'cv(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta = 100$, $\alpha = 1\%$) (cf. (B)-3). Les trajectoires $(t', I'cv(t, t'))_i$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t = Tcv - 1$, Tcv et $Tcv + 1$, où $Tcv = 6315$ itérations qui sont données par la relation (B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t = Tcv - 1$ ne vérifie pas la condition de la relation (B)-3). Donc, la trajectoire associée à la valeur $t = Tcv$ est bien minimale au sens de la relation (B)-3).

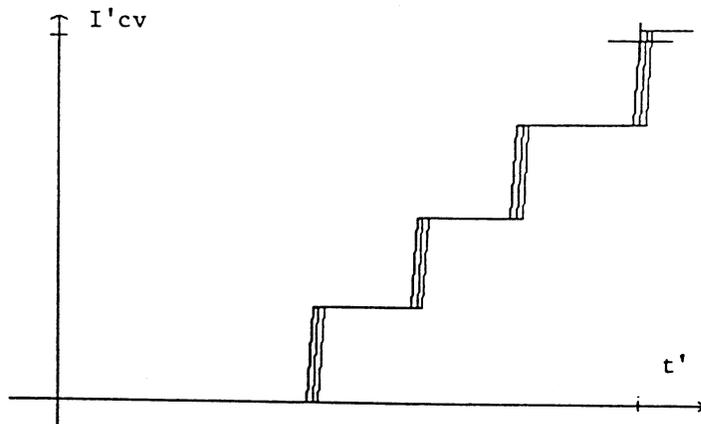


Figure 56 : Réseau de neurones de taille $N=128 \times 128$ en mode d'implémentation bloc-séquentiel. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec t fixé (cf. (B)-1). La croix représente les coordonnées du point critique ($\delta=100, \alpha=1\%$)(cf. (B)-3). Les trajectoires $(t', I_{cv}(t, t'))_t$ sont quasiment décroissantes quand t croît. Ces courbes correspondent respectivement aux valeurs de $t=T_{cv}-1, T_{cv}$ et $T_{cv}+1$, où $T_{cv}=5159$ itérations qui sont données par la relation ((B)-3) (cf. Figure 58). La trajectoire qui correspond à la valeur de $t=T_{cv}-1$ ne vérifie pas la condition de la relation ((B)-3). Donc, la trajectoire associée à la valeur $t = T_{cv}$ est bien minimale au sens de la relation ((B)-3).

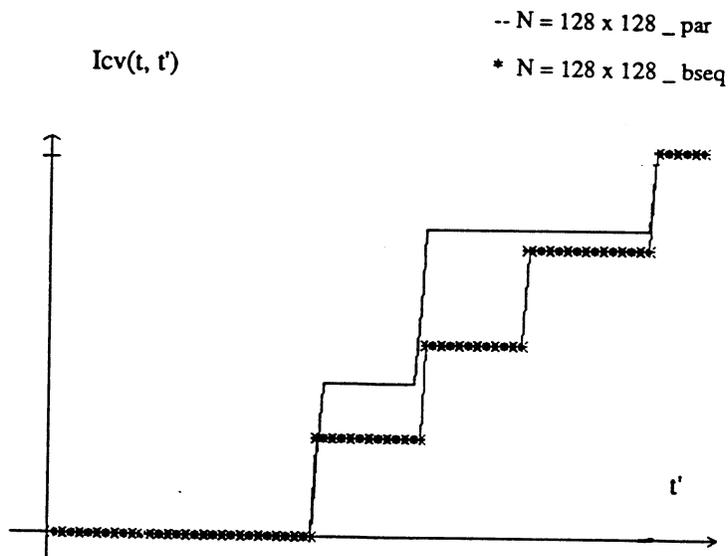


Figure 57 : Réseau de neurones de taille $N = 128 \times 128$. Evolution de l'indice de convergence $I_{cv}(t, t')$ en fonction de t' , avec $t = T_{cvs}$ (correspond au réseau en mode d'implémentation bloc-séquentiel) et $t=T_{cvp}$ (correspond au réseau en mode d'implémentation parallèle) fixés (cf. (B)-1).

Enfin, le temps de convergence du système, noté par T_{cv} (cf. Figure 58), est calculé par :

$$T_{cv} = \inf_{t \geq 1} \{ A_{\alpha, \delta}(t, t') \}, \quad [(B)-3]$$

où $A_{\alpha, \delta}(t, t') = \{ t \geq 1 ; \forall t' \leq \delta, I_{cv}(t, t') \leq \alpha \}$

Application :

(cf. Figure 58) le temps de convergence (T_{cv}), en unité d'itérations, correspond aux paramètres : $\alpha = 1\%$ et $\delta = 100$.

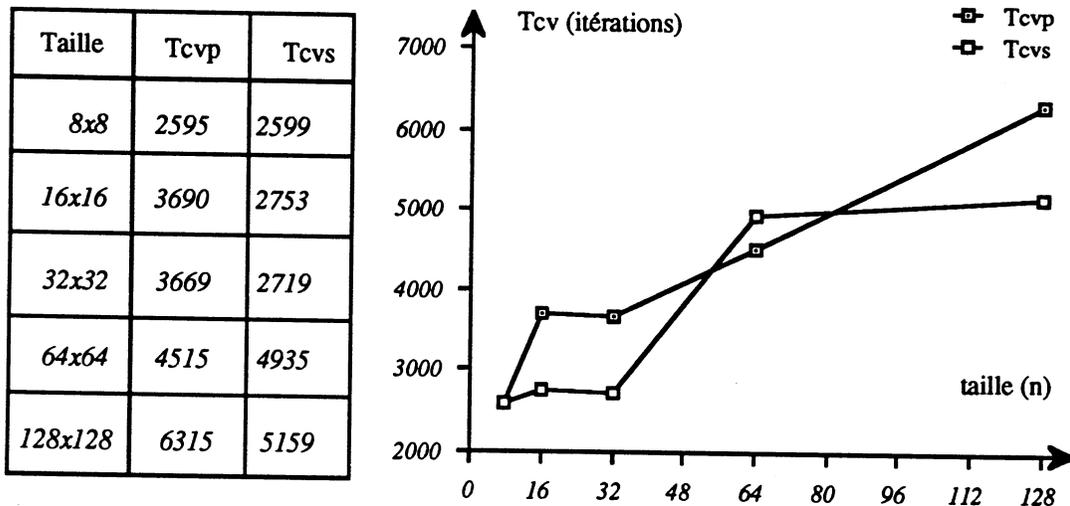


Figure 58 : Temps de convergence des réseaux en modes parallèle et bloc-séquentiel

En conclusion, comme ci-dessus, les temps de convergence sont plus importants en mode parallèle ; ceci correspond à une augmentation plus rapide de l'indice de convergence en mode séquentiel.

IV -3.2.3 Visualisation des potentiels d'interaction (singletons et paires)

Selon les tailles des réseaux, on a visualisé les potentiels d'interaction, singletons et paires, en fonction de leurs valeurs. En effet, les potentiels d'interaction de singletons sont représentés par des carrés pleins de dimension décroissante, i.e. la dimension du carré est proportionnelle à la valeur du potentiel en ce site, et les potentiels d'interaction de paires sont représentés par des barres (ou des segments). Dans la représentation des potentiels d'interaction de paires, on les a, dans le premier cas, représentés par des barres de surface décroissante, i.e. la surface de la barre, est proportionnelle à la valeurs du potentiel en le couple de sites (i, j) et, dans le deuxième cas, on a transféré les images (singletons et paires) sur un ordinateur spécialisé dans le traitement d'images (VINIX), pour représenter les potentiels d'interaction de paires en segments simples, mais en fausse couleur. On n'a pas présenté les images du premier cas dans ce manuscrit, et cela pour des raisons de visibilité, i.e. on ne voyait plus les représentations des potentiels de singletons, représentés par des carrés pleins. Par contre, on a présenté que quelques photos des potentiels d'interaction de singletons et de paires en fausse couleur (cf. Annexe 2), les autres images étant présentées en noir et blanc (mais pas en photos). Les valeurs des potentiels d'interaction de singletons et de paires sont données par, qui sont notées respectivement par : J^*_S , J^*_P , sont données par :

$$J^*_S(\{i\}) = \log \left[\frac{\text{Nbre de } \{i\} \text{ apparatus}}{\text{Nbre de } \emptyset \text{ apparatus}} \right],$$

$$J^*_P(\{i, j\}) = -(J^*_S(\{i\}) + J^*_S(\{j\})) + \log \left[\frac{\text{Nbre de } \{i, j\} \text{ apparatus}}{\text{Nbre de } \emptyset \text{ apparatus}} \right].$$

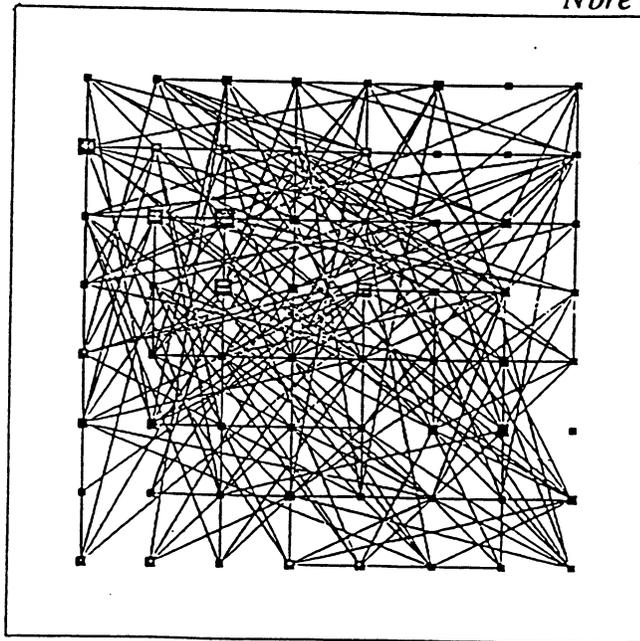


Figure 59: Réseau de neurones de taille $N=8 \times 8$ en mode d'itération parallèle. Evolution pendant (12000-... Tcvp) itérations (cf. Figure 58). Représentation des potentiels d'interaction de singletons par des carrés pleins (resp. de paires par des barres). Ces potentiels sont normalisés. L'intensité des potentiels d'interaction de singletons est proportionnelle à la taille des carrés pleins ainsi représentés.

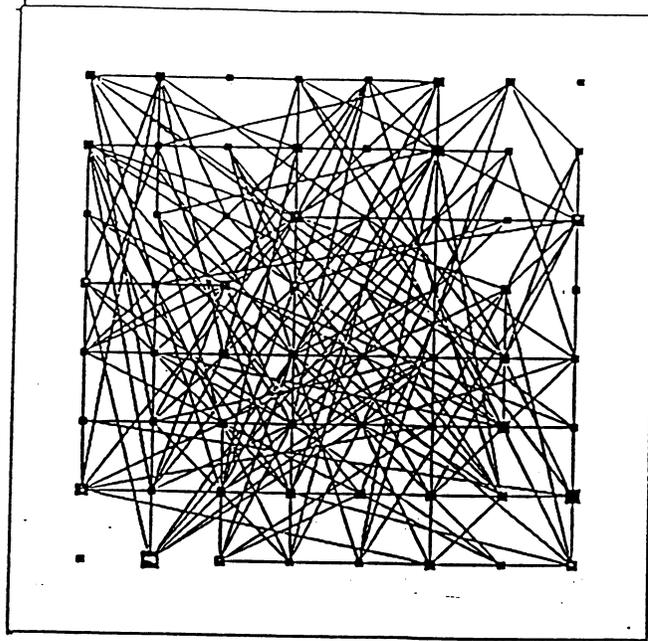


Figure 60 : Réseau de neurones de taille $N=8 \times 8$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des potentiels d'interaction de singletons par des carrés pleins (resp. de paires par des segments). Ces potentiels sont normalisés. L'intensité des potentiels d'interaction de singletons est proportionnelle à la taille des carrés pleins ainsi représentés.

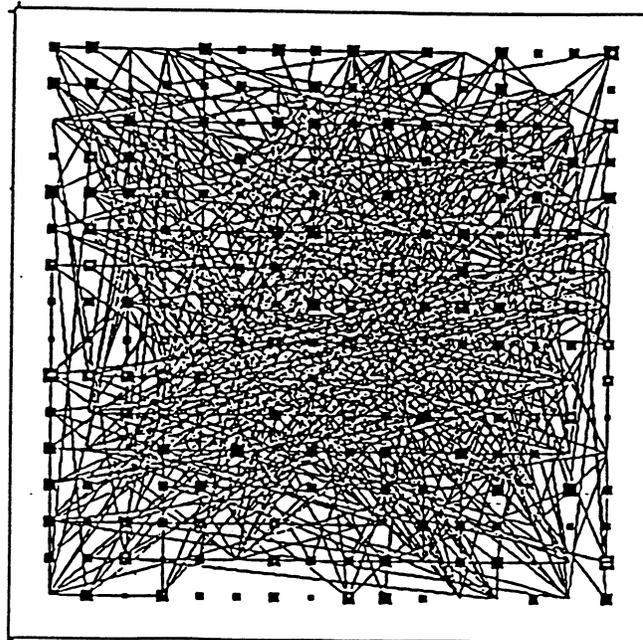


Figure 61 : Réseau de neurones de taille $N=16 \times 16$ en mode d'itération parallèle. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des potentiels d'interaction de singletons par des carrés pleins (resp. de paires par des segments). Ces potentiels sont normalisés. L'intensité des potentiels d'interaction de singletons est proportionnelle à la taille des carrés pleins ainsi représentés.

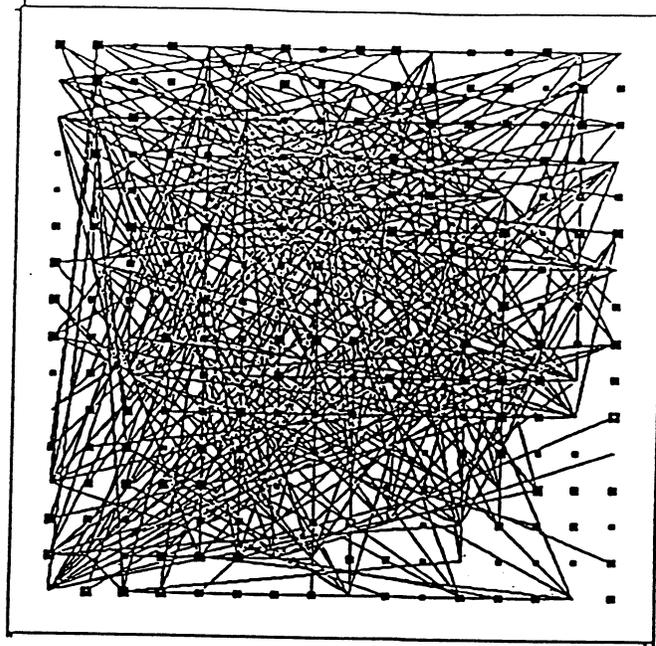


Figure 62 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des potentiels d'interaction de singletons par des carrés pleins (resp. de paires par des segments). Ces potentiels sont normalisés. L'intensité des potentiels d'interaction de singletons est proportionnelle à la taille des carrés pleins ainsi représentés.

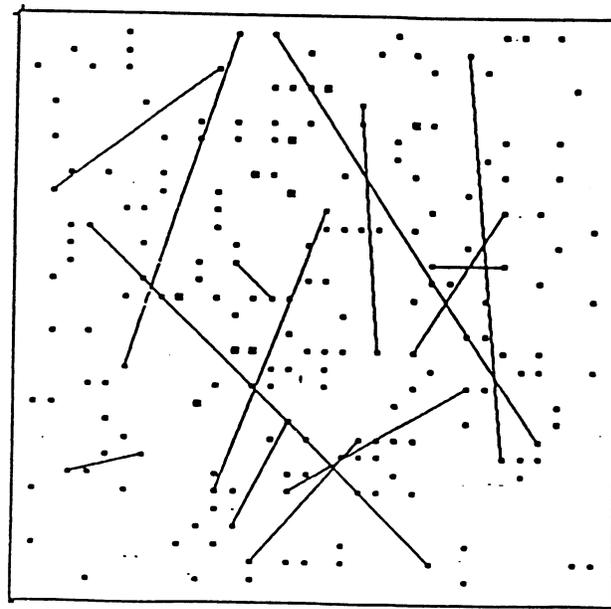


Figure 63 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des potentiels d'interaction de singletons par des carrés pleins (resp. de paires par des segments). Ces potentiels sont normalisés. L'intensité des potentiels d'interaction de singletons est proportionnelle à la taille des carrés pleins ainsi représentés.

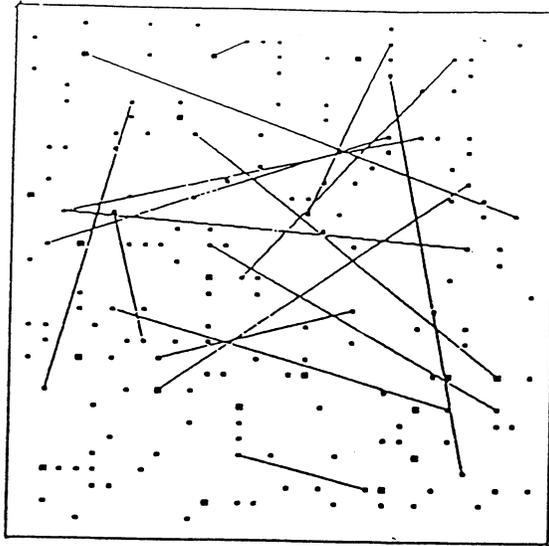


Figure 66 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des potentiels d'interaction de singletons par des carrés pleins (resp. de paires par des segments). Ces potentiels sont normalisés. L'intensité des potentiels d'interaction de singletons est proportionnelle à la taille des carrés pleins ainsi représentés.

IV -3.2.4 Cliques et Amas du réseau

Dans ce paragraphe, nous allons définir deux types de caractères spatiaux du réseau selon la topologie des voisinages.

Définition : Cliques

Soit V un système de voisinages sur Ω , la configuration "C" de $P(\Omega)$ est dite configuration clique relativement à V , si pour tout couple de sites (i, j) de "C" avec $i \neq j$, alors "i" et "j" sont voisins au sens V .

Définition : Amas

Soit V un système de voisinages sur Ω , la configuration "C" de $P(\Omega)$ est dite configuration amas relativement à V , si pour tout site "i" de "C", il existe au moins un site "j" de "C", où $j \neq i$, tel que "i" et "j" sont voisins au sens V .

Remarque : Toute configuration clique est une configuration amas. L'inverse est faux, par définition même.

Applications :

On a choisi la topologie de voisinages : $V = \text{Sphères de Manhattan}$ (cf. § II -5.3.2). Dans un réseau de N sites, du fait de cette topologie, on ne peut avoir que 4 types de cliques, c'est à dire des cliques d'ordres : 1, 2, 3 et 4, qui sont notées respectivement par : 1-cliques, 2-cliques, 3-cliques et 4-cliques.

Par exemple si $N=64$ sites, alors au total il y a (64×1) configurations 1-cliques, (64×6) configurations 2-cliques, (64×10) configurations 3-cliques et (64×6) configurations 4-cliques.

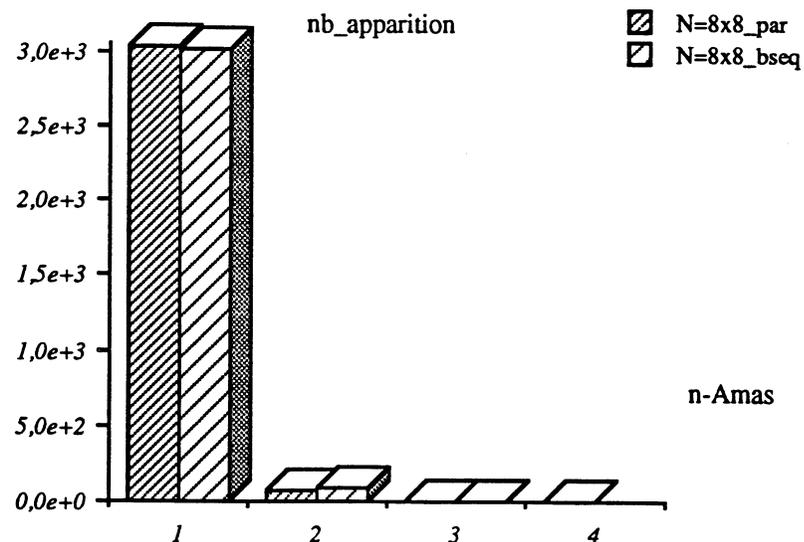
(A) - Histogrammes d'évolution

• Réseau à comportement parallèle de taille $N=8 \times 8$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	3040	1	3040
2	75	2	75
3	1	3	4
4	0	4	1

• Réseau à comportement bloc-séquentiel de taille $N=8 \times 8$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	3023	1	3023
2	87	2	87
3	2	3	4
4	0	4	0

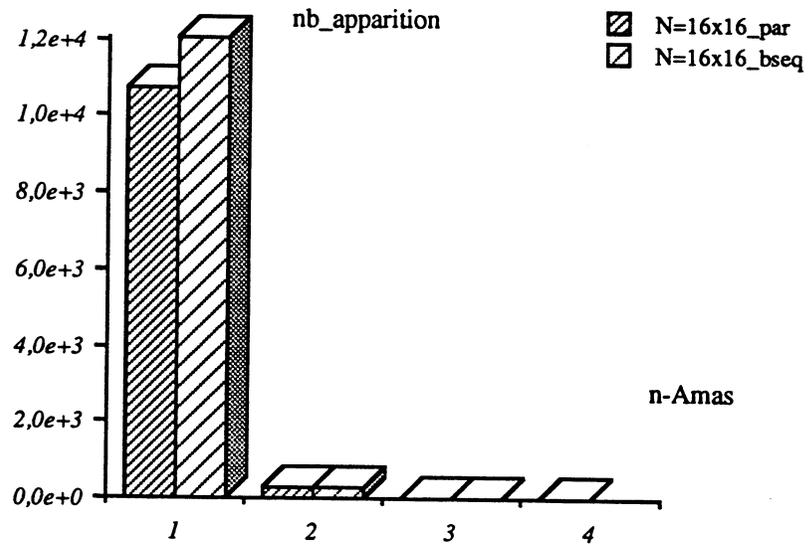


• Réseau à comportement parallèle de taille $N=16 \times 16$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	10736	1	10736
2	309	2	309
3	3	3	6
4	0	4	1

• Réseau à comportement bloc-séquentiel de taille $N=16 \times 16$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	12045	1	12045
2	298	2	298
3	4	3	13
4	0	4	0

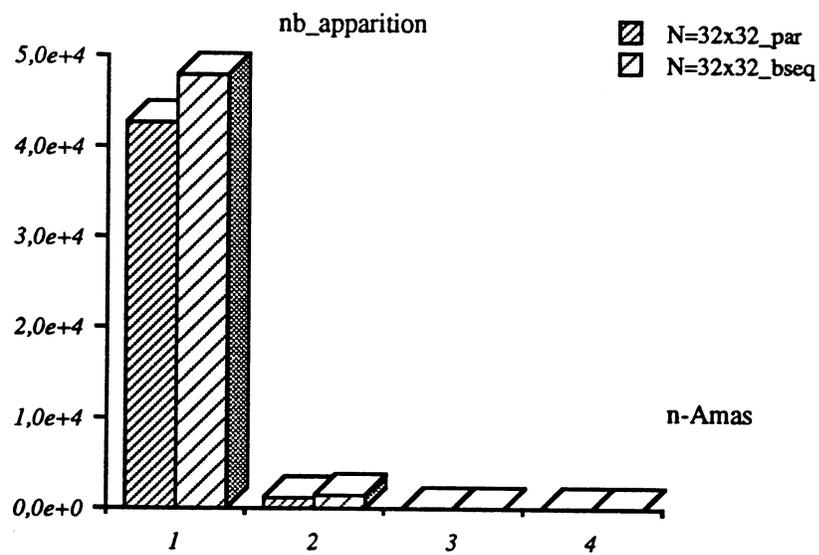


• Réseau à comportement parallèle de taille $N=32 \times 32$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	42724	1	42724
2	1299	2	1299
3	9	3	49
4	0	4	2

• Réseau à comportement bloc-séquentiel de taille $N=32 \times 32$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	47813	1	47813
2	1482	2	1482
3	11	3	64
4	0	4	1

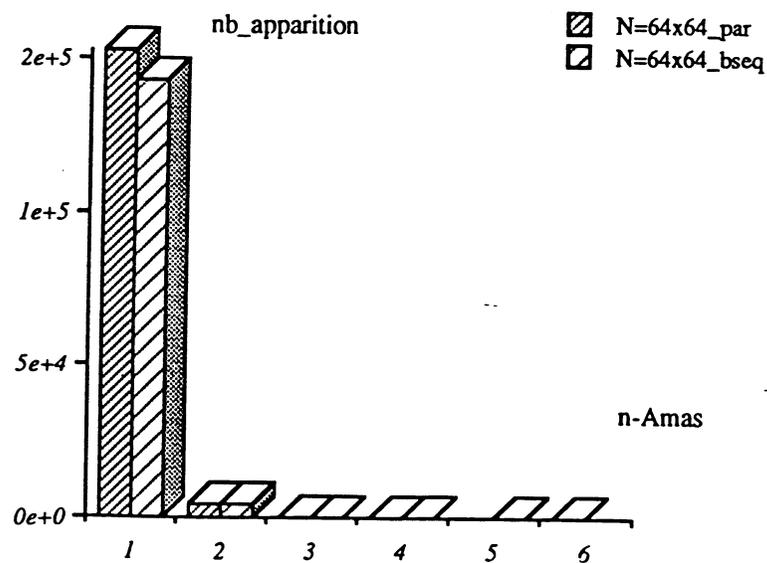


• Réseau à comportement parallèle de taille $N=64 \times 64$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	152564	1	152564
2	4606	2	4606
3	39	3	182
4	0	4	7
5	0	5	0
6	0	6	1

• Réseau à comportement bloc-séquentiel de taille $N=64 \times 64$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	143745	1	143745
2	4422	2	4422
3	37	3	186
4	1	4	9
5	0	5	2
6	0	6	0

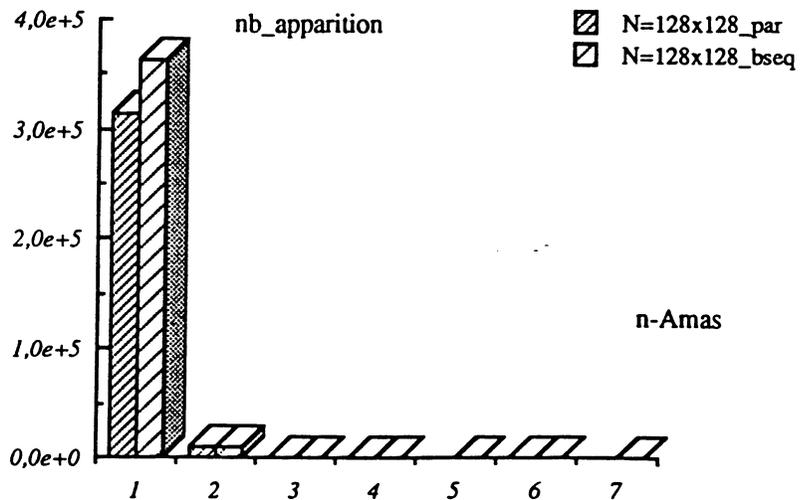


• Réseau à comportement parallèle de taille $N=128 \times 128$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	314702	1	314702
2	9643	2	9643
3	75	3	346
4	0	4	18
5	0	5	0
6	0	6	8

• Réseau à comportement bloc-séquentiel de taille $N=128 \times 128$

n-cliques (n)	nb_appar	n-amas (n)	nb_appar
1	363032	1	363032
2	10822	2	10822
3	103	3	411
4	0	4	16
5	0	5	1
6	0	6	5
7	0	7	1
8	0	8	1
9	0	9	2



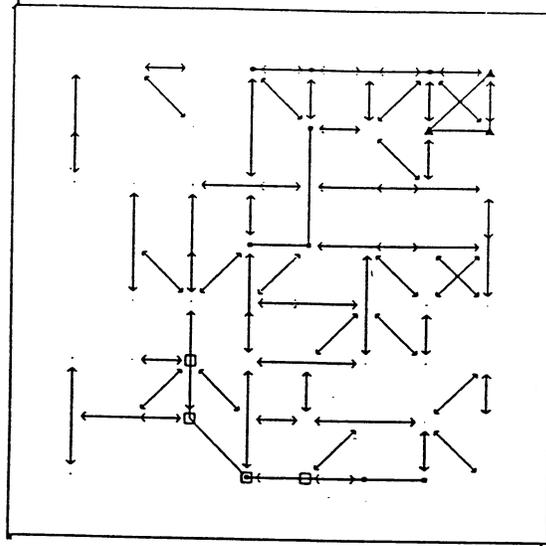
(B) - Visualisation des Cliques et Amas

Figure 65 : Réseau de neurones de taille $N = 8 \times 8$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (75) 2-cliques en flèches double sens (y compris les identiques), (1) 3-cliques en triangles pleins, (3) 3-amas (non cliques) en carrés pleins et (1) 4-amas (non cliques) en carré vide.

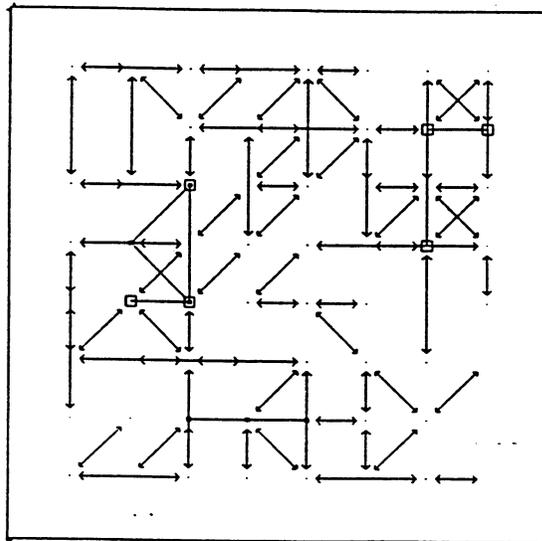


Figure 66 : Réseau de neurones de taille $N = 8 \times 8$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (87) 2-cliques en flèches double sens (y compris les identiques), (2) 3-cliques en carrés pleins et (2) 3-amas (non cliques) en carrés vides.

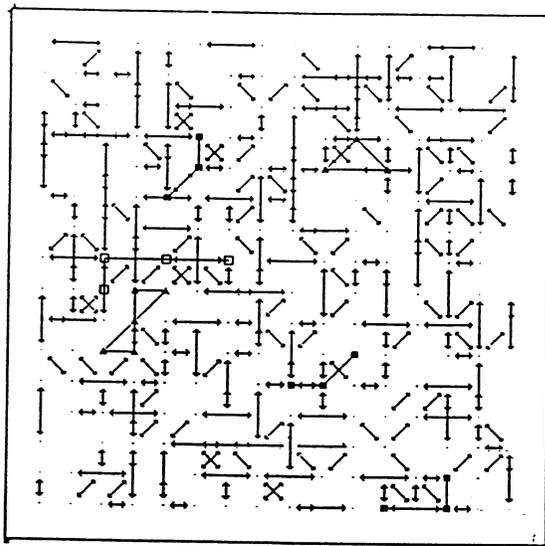


Figure 67 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (309) 2-cliques en flèches double sens (y compris les identiques), (3) 3-cliques en triangle pleins, (3) 3-amas (non cliques) en carrés pleins et (1) 4-amas (non cliques) en carré vide.

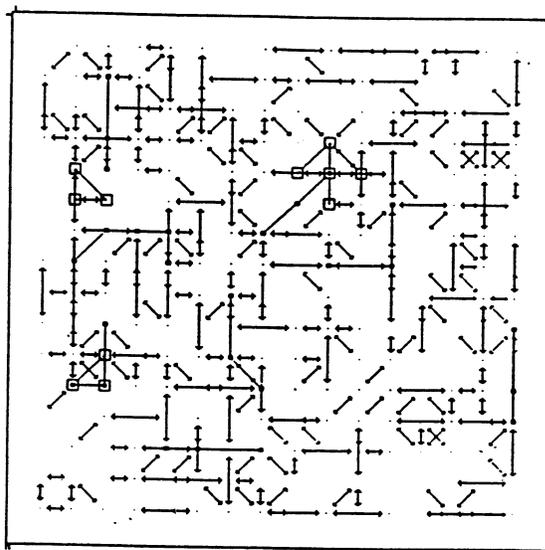


Figure 68 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (298) 2-cliques en flèches double sens (y compris les identiques), (4) 3-cliques en carrés vides et (9) 3-amas (non cliques) en carrés pleins.

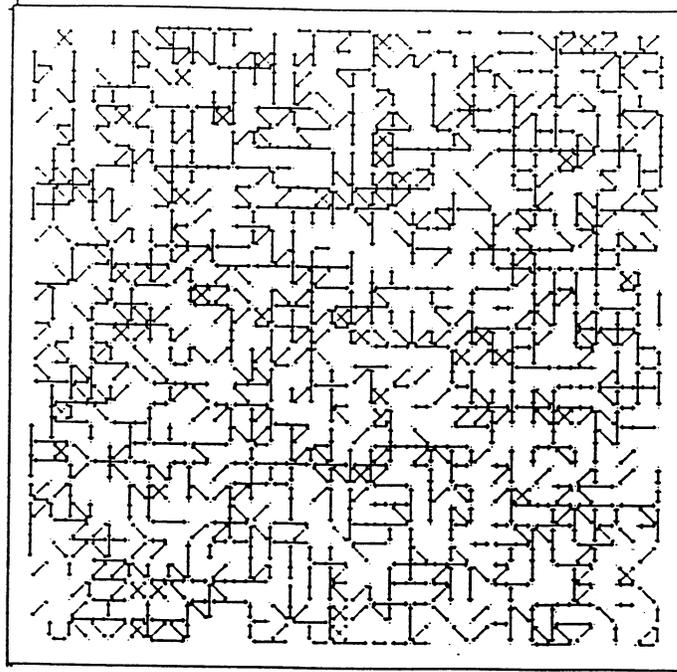


Figure 69 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des configurations (1299) 2-cliques en flèches double sens (y compris les identiques).

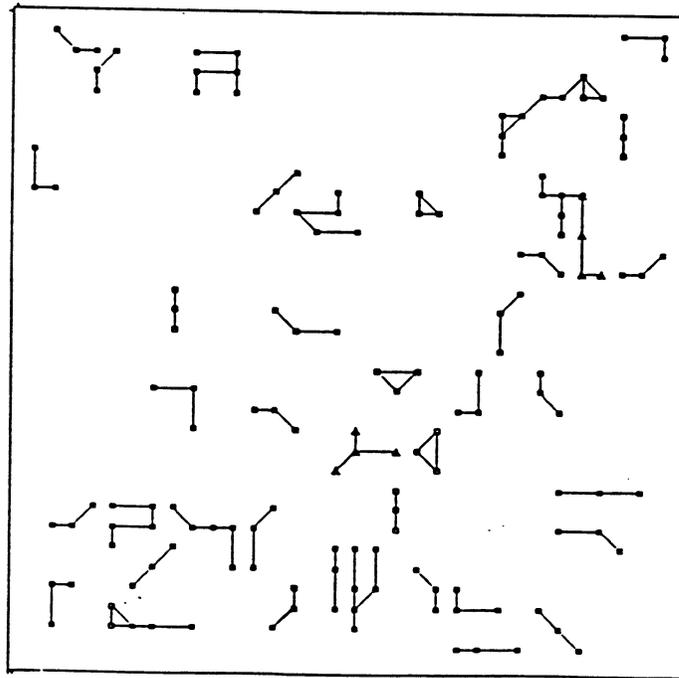


Figure 70 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (9) 3-cliques en carrés vides et (40) 3-amas (non cliques) en carrés pleins et (2) 4-amas (non cliques) en triangles pleins.

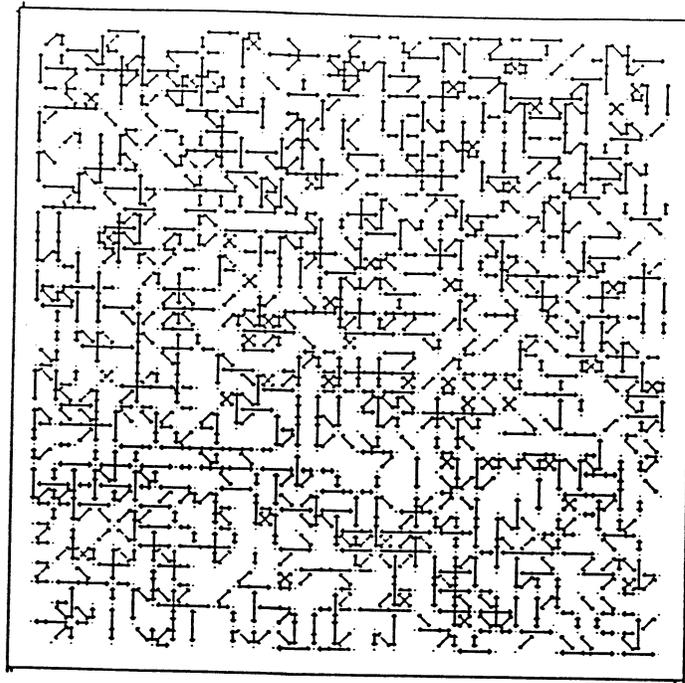


Figure 71 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des configurations (1482) 2-cliques en flèches double sens (y compris les identiques).

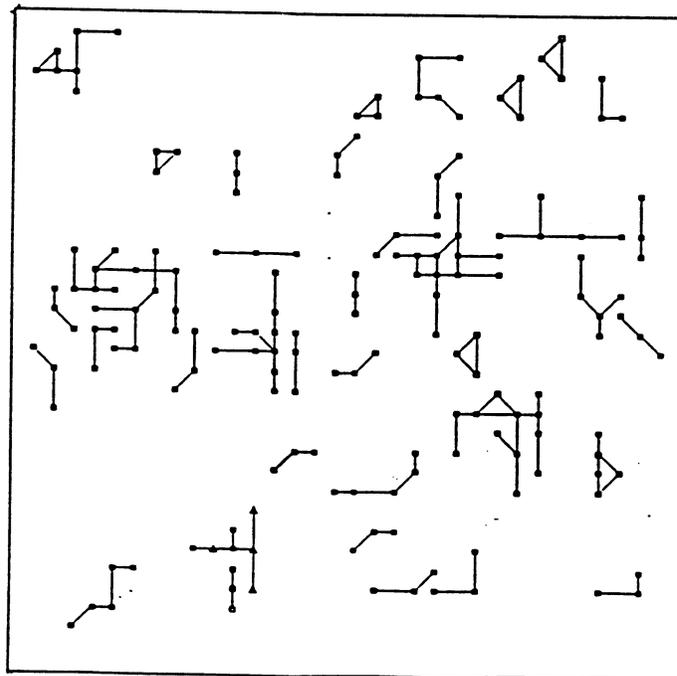


Figure 72 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (11) 3-cliques en carrés vides et (53) 3-amas (non cliques) en carrés pleins et (1) 4-amas (non cliques) en triangle plein.

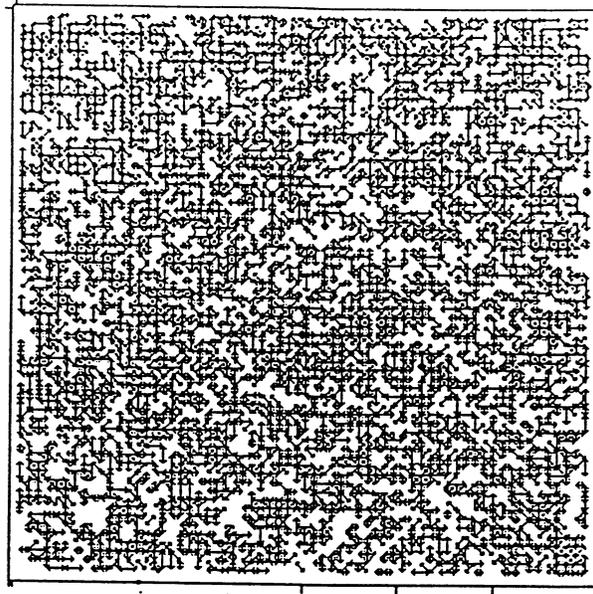


Figure 73 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des configurations (4606) 2-cliques en flèches double sens (y compris les identiques).

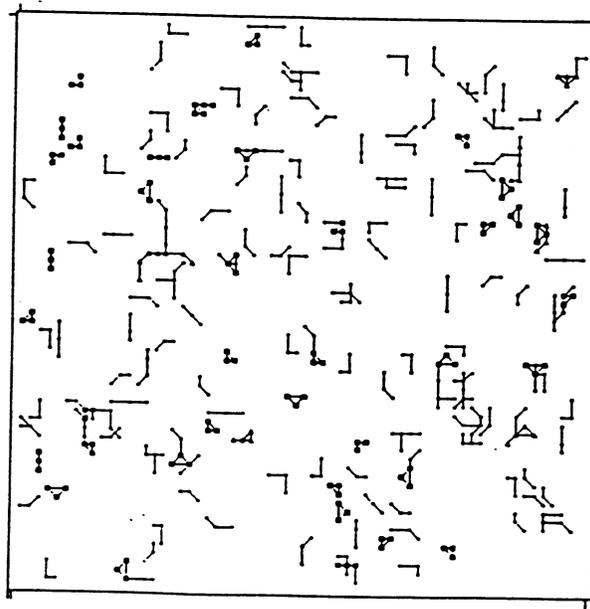


Figure 74 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (39) 3-cliques en carrés vides et (143) 3-amas (non cliques) en carrés pleins, (1) 4-cliques en triangles vides et (6) 4-amas (non cliques) en triangles pleins.

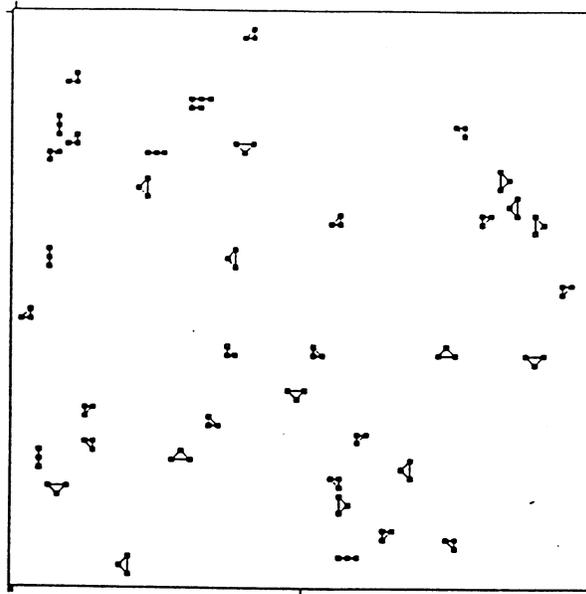


Figure 75 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (39) 3-cliques en carrés vides.

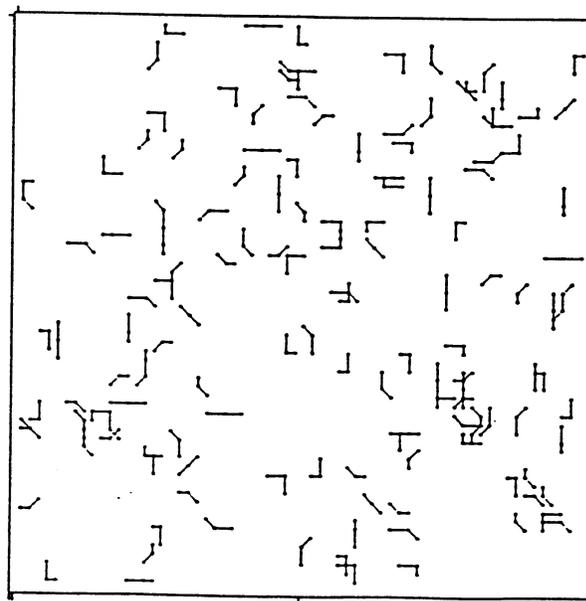


Figure 76 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (143) 3-amas (non cliques) en carrés pleins.

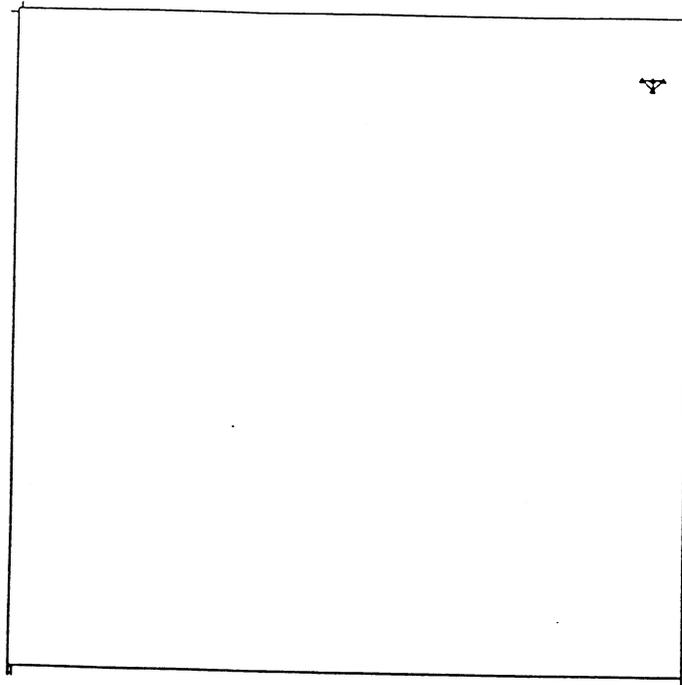


Figure 77 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (1) 4-cliques en triangle vide.

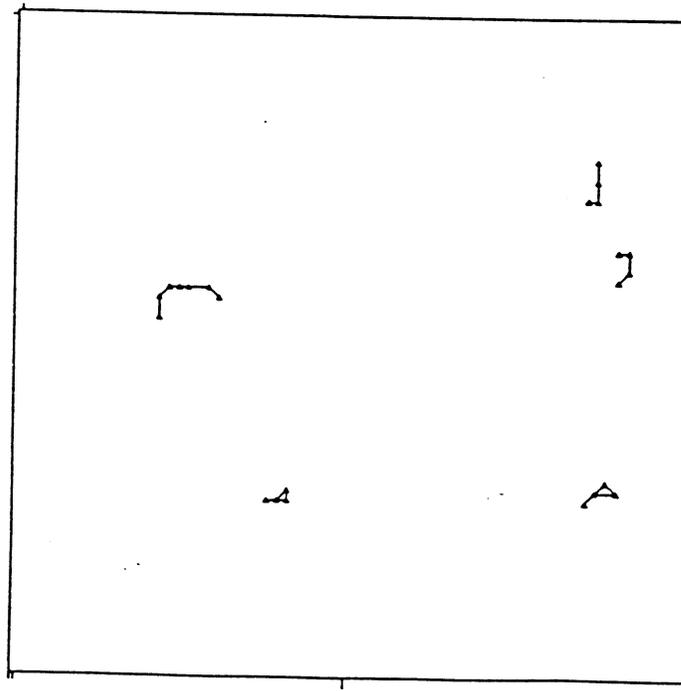


Figure 78 : Réseau de neurones de taille $N=64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (6) 4-amas (non cliques) en triangles pleins.

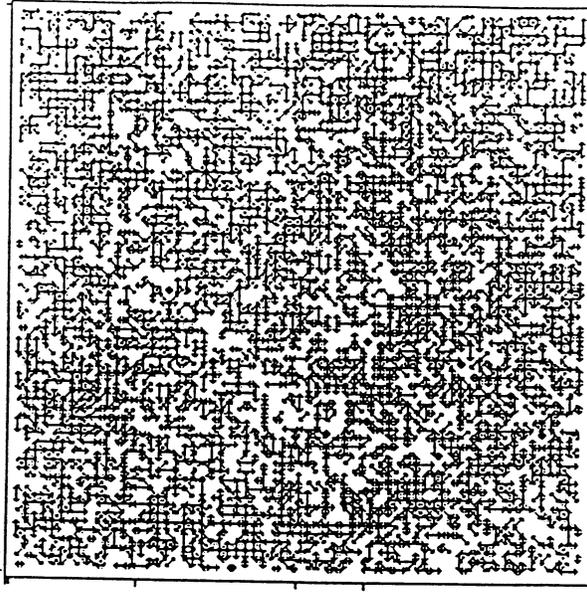


Figure 79 : Réseau de neurones de taille $N=64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant $(12000-T_{cvs})$ itérations (cf. Figure 58). Représentation des configurations (4422) 2-cliques en flèches double sens (y compris les identiques).

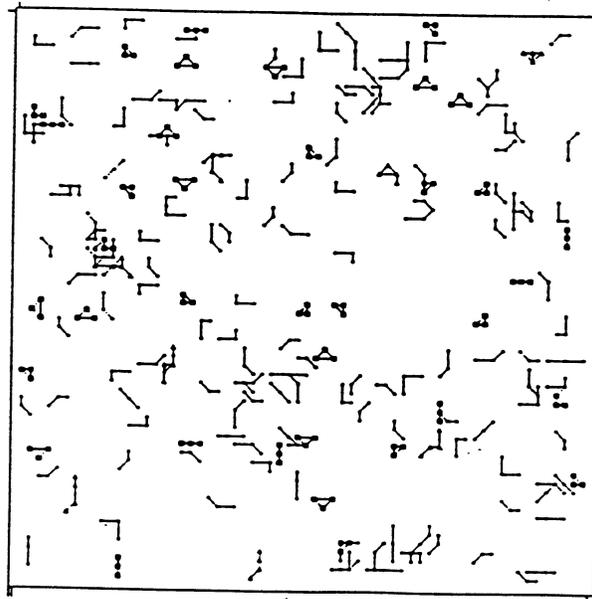


Figure 80 : Réseau de neurones de taille $N=64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant $(12000-T_{cvs})$ itérations (cf. Figure 58). Représentation simultanée des configurations suivantes : (37) 3-cliques en carrés vides et (149) 3-amas (non cliques) en carrés pleins, (1) 4-cliques en triangles vides et (8) 4-amas (non cliques) en triangles pleins.

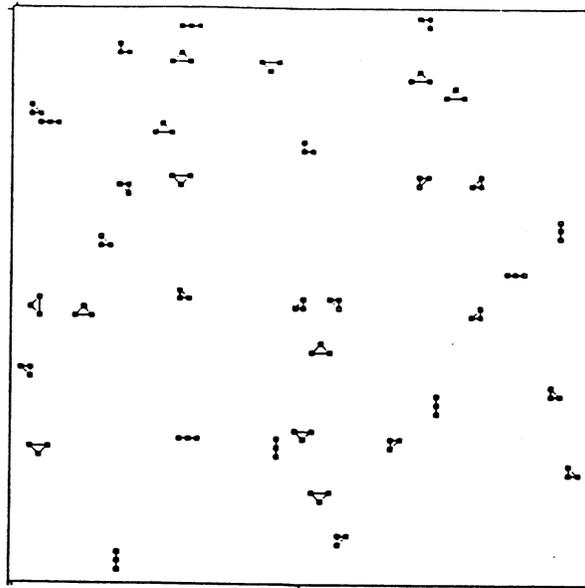


Figure 81 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (37) 3-cliques en carrés vides.

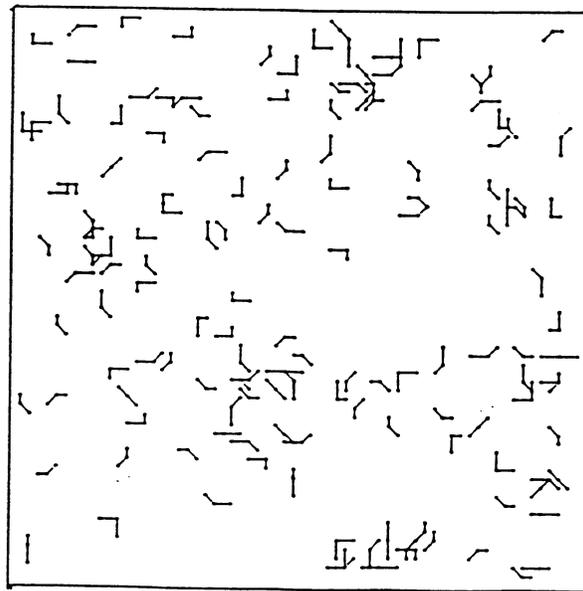


Figure 82 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (149) 3-amas (non cliques) en carrés pleins.

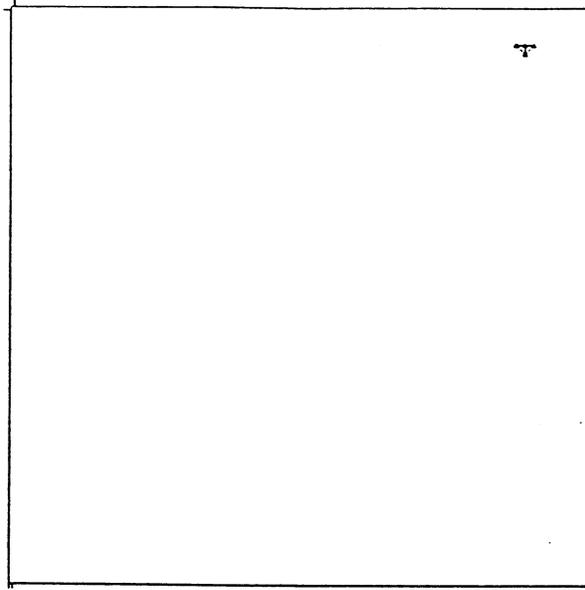


Figure 83 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (1) 4-cliques en triangle vide.

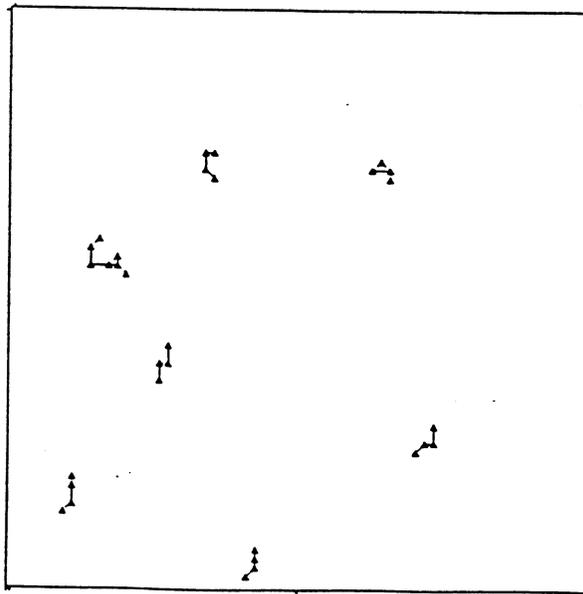


Figure 84 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (8) 4-amas (non cliques) en triangles pleins.

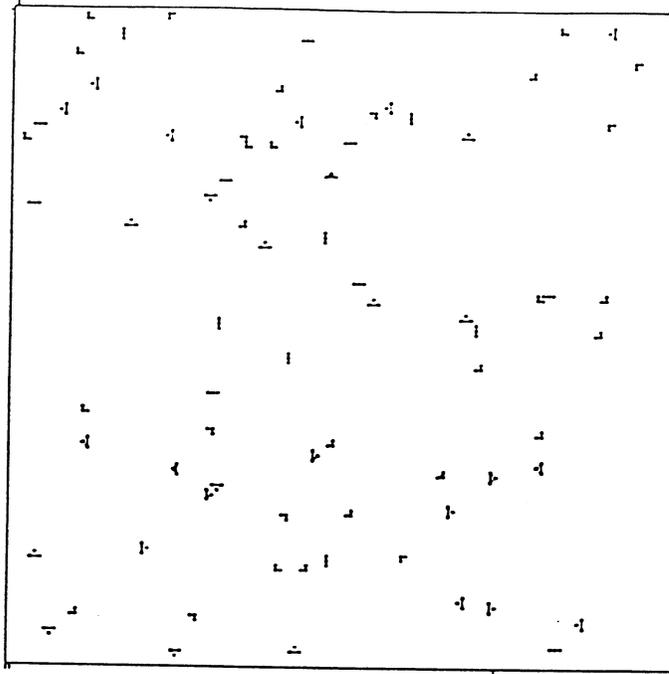


Figure 85 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'itération parallèle. Evolution pendant (10200-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (75) 3-cliques en carrés pleins.

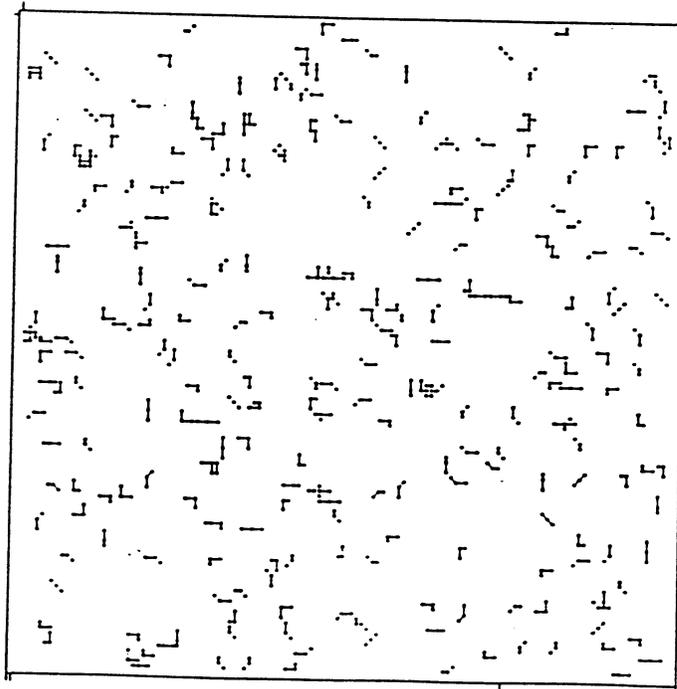


Figure 86 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'itération parallèle. Evolution pendant (10200-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (271) 3-amas (non cliques) en carrés pleins.

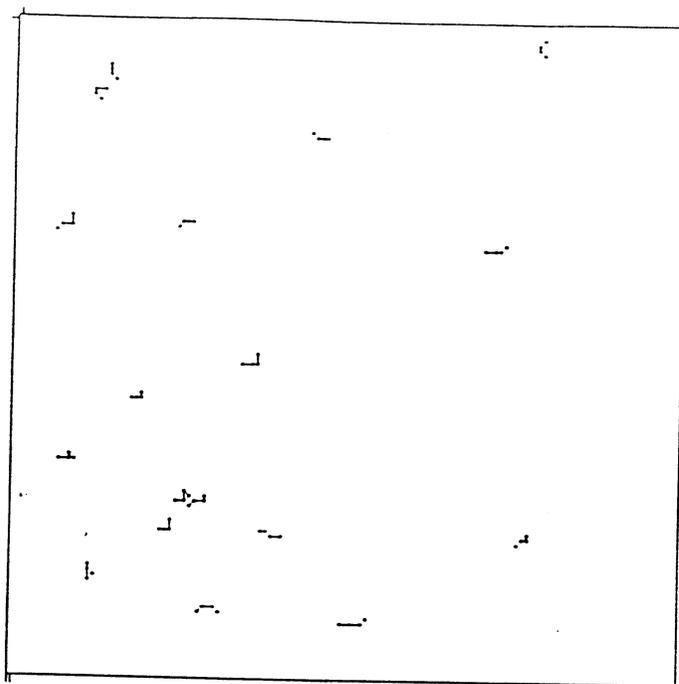


Figure 87 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'itération parallèle. Evolution pendant (10200-Tcvp) itérations (cf. Figure 58). Représentation des configurations : (18) 4-amas (non cliques) en carrés pleins.

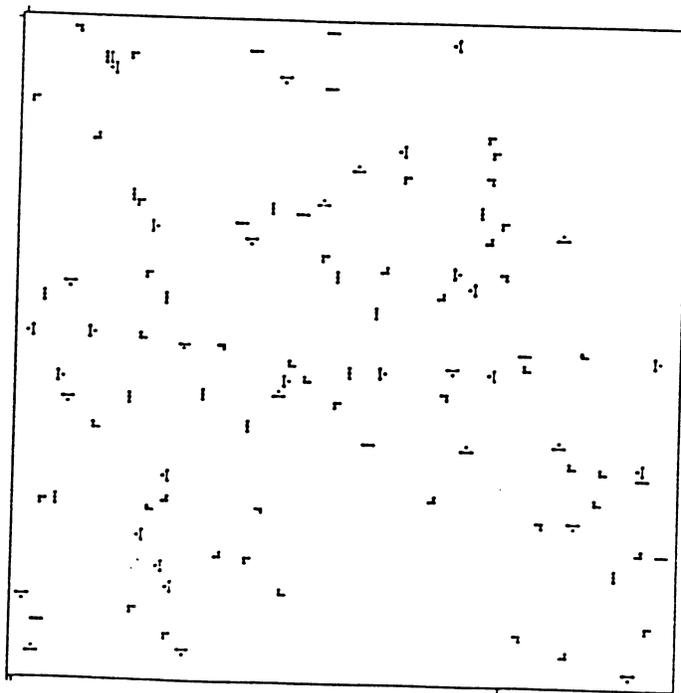


Figure 88 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'itération bloc-séquentiel. Evolution pendant (10200-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (103) 3-cliques carrés pleins.

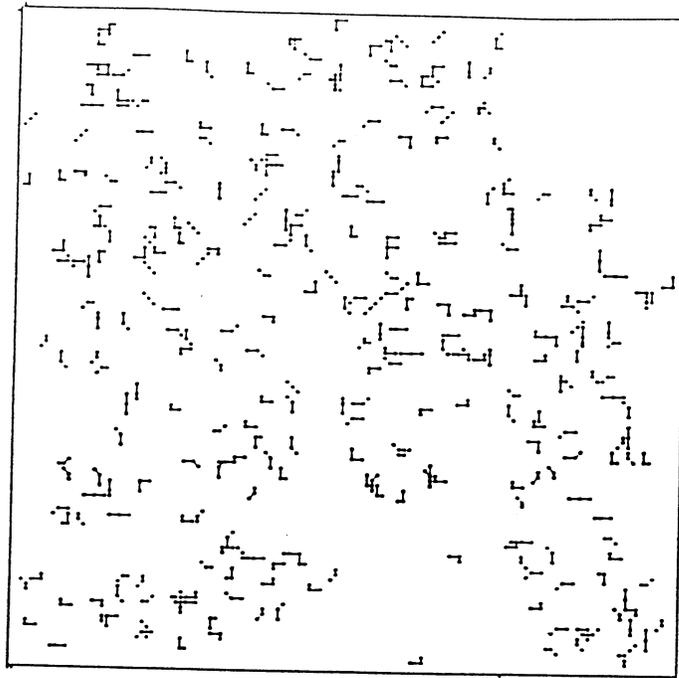


Figure 89 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'itération bloc-séquentiel. Evolution pendant (10200-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (308) 3-amas (non cliques) en carrés pleins.

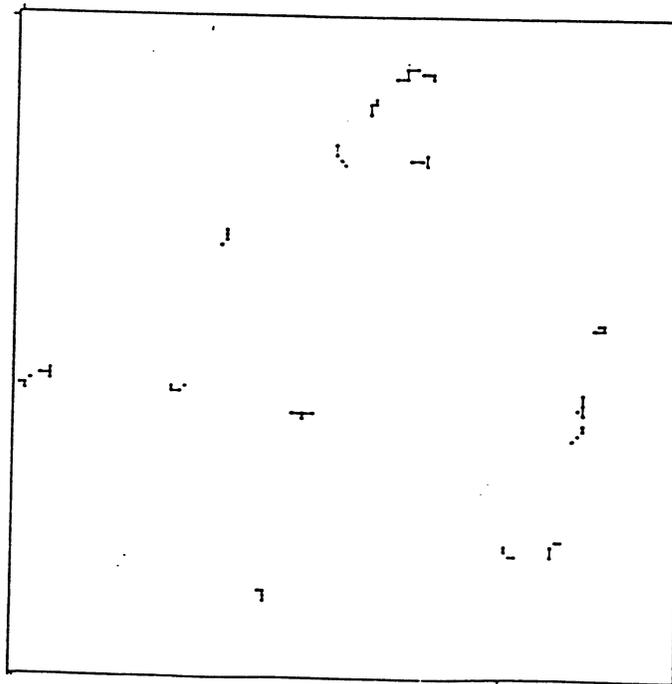


Figure 90 : Réseau de neurones de taille $N = 128 \times 128$ en mode d'itération bloc-séquentiel. Evolution pendant (10200-Tcvs) itérations (cf. Figure 58). Représentation des configurations : (16) 4-amas en carrés pleins.

IV -3.2.4 Tests de validation de la mesure de Gibbs

soit μ_{emp} , la mesure empirique du réseau, telle que : $\mu_{emp}(A) = (\text{nombre de } A \text{ apparus})/K$, où K est l'intervalle du temps pendant lequel le réseau évolue. A partir de cette mesure empirique, on calcule, par exemple, les potentiels d'interaction de singletons, de paires et de triplets, notés respectivement par : J^*_S , J^*_P et J^*_T , tels que :

$$J^*_S(\{i\}) = \log \left[\frac{\mu_{emp}(\{i\})}{\mu_{emp}(\emptyset)} \right],$$

$$J^*_P(\{i, j\}) = -(J^*_S(\{i\}) + J^*_S(\{j\})) + \log \left[\frac{\mu_{emp}(\{i, j\})}{\mu_{emp}(\emptyset)} \right],$$

$$J^*_T(\{i, j, k\}) = +(J^*_S(\{i\}) + J^*_S(\{j\}) + J^*_S(\{k\}))$$

$$- (J^*_P(\{i, j\}) + J^*_P(\{i, k\}) + J^*_P(\{k, j\}))$$

$$+ \log \left[\frac{\mu_{emp}(\{i, j, k\})}{\mu_{emp}(\emptyset)} \right].$$

Cela nous permet de calculer la mesure μ_{SP} (la mesure reconstituée seulement par des potentiels d'interaction de singletons et de paires) sur les configurations triplets (ou 3-uplets), telle que :

$$\mu_{SP}(\{i, j, k\}) = P_0 \text{Exp} \left[\sum_{A \subseteq \{i, j\}} J^*_U(A) \right], \text{ où } P_0 = \mu_{emp}(\emptyset)$$

et où $J^*_U(A)$ est le potentiel d'interaction empirique de la configuration "A".

- Le premier test consiste à comparer les deux mesures μ_{emp} et μ_{SP} sur les configurations 3-uplets à l'aide d'une distance appelée : la distance de "Kullback", telle que :

$$dkull(\mu^*_{emp}, \mu^*_{SP}) = \sum_{3\text{-uplets}} [\mu^*_{emp}(\{i, j, k\}) - \mu^*_{SP}(\{i, j, k\})] \log \left[\frac{\mu^*_{emp}(\{i, j, k\})}{\mu^*_{SP}(\{i, j, k\})} \right]$$

où μ^*_{emp} et μ^*_{SP} sont respectivement des mesures normalisées.

- Le second test consiste à calculer les potentiels de singletons en cliques, ainsi que leur moyenne, et ensuite les comparer aux anciens.

Applications :

Ces expériences ont été faites en deux modes d'implémentation différents (parallèle, bloc-séquentiel). Elles sont résumées dans les tableaux suivants :

D'abord expliquons les éléments des tableaux en questions :

- * *taille* : est la taille du réseau,
- * *Kmax* : est le nombre maximum d'itérations,
- * *Tcv* : est le temps de convergence (ou début du régime stationnaire),
- * *Ntrip* : est le nombre de configurations triplets pendant (*Kmax* - *Tcv*) itérations,
- * *Nvide* : est le nombre de configurations vides pendant (*Kmax* - *Tcv*) itérations,
- * $P_0 = \mu_{emp}(\emptyset) = (\text{nombre de } \emptyset \text{ apparus}) / (Kmax - Tcv),$
- * $\bar{\mu}_{emp} = \frac{1}{Ntrip} \sum_{\text{triplets}} \mu_{emp}(A),$
- * $\bar{\mu}_{s,p} = \frac{1}{Ntrip} \sum_{\text{triplets}} \mu_{s,p}(A),$
- * *Jsmín* : est le minimum des potentiels d'interaction de singletons pendant *Kmax* itérations,
- * *Jsmáx* : est le maximum des potentiels d'interaction de singletons pendant *Kmax* itérations,
- * *Jsmín_c* : est le minimum des potentiels d'interaction de singletons en cliques pendant *Kmax* itérations,
- * *Jsmáx_cl* : est le maximum des potentiels d'interaction de singletons en cliques pendant *Kmax* itérations,
- * *Ordre* : est la valeur de rééchantillonnage, i.e. rééchantillonner les "*n-cliques*" jusqu' à la configuration "*Ordre-uplets*".

*** Résultats du test N° 1 :**

taille	8x8	16x16	32x32
Kmax	15000	15000	25000
Tcv	2595	3690	2979
Ntrip	55	1216	2647
Nvide	8803	2846	118
P ₀	7.10 E-01	2.52 E-01	5.36 E-03
$\bar{\mu}_{emp}$	8.07 E-05	8.83 E-05	4.54 E-03
$\bar{\mu}_{s,p}$	8.21 E-05	8.84 E-05	4.54 E-05
dkull	1.52 E-01	1.87 E-01	1.29 E-01

Tableau 14 : Réseau à comportement parallèle

*** Résultats du test N°2 :**

taille	8x8	16x16	32x32
Kmax	15000	15000	15000
Ordre	10	10	20
Jmin	-9.0613	-8.0567	-4.5218
Jmax	-5.0910	-4.7986	-3.4232
Jmin_cl	-11.662	-11.556	-12.091
Jmax_cl	-7.4722	-7.3515	-7.8564

Tableau 15 : Réseau à comportement parallèle

taille	8x8	16x16	32x32
Kmax	15000	15000	15000
Ordre	10	10	20
Jmin	-9.0604	-8.0774	-4.5951
Jmax	-5.0531	-4.7816	-3.4965
Jmin_cl	-11.662	-11.556	-12.090
Jmax_cl	-7.4427	-7.3667	-7.8415

Tableau 16 : Réseau à comportement bloc-séquentiel

Remarque : Pour les tailles 64 x 64 et 128 x 128, on a de faibles apparitions de singletons, sans échantillonnage par cliques (cf. IV -3.2.1 § 1). Pour cette raison, les comparaisons, dans le cas de ces tailles, n'ont pas eu lieu. Par exemple dans le cas du réseau de taille 32x32, on a observé 311 configurations singletons (sans cliques) contre 1024 de singletons (avec cliques), d'où l'écart entre ces deux types d'échantillonnages.

En conclusion, nous observons que la restructurabilité de la mesure de Gibbs à partir uniquement de ses potentiels de singletons et de paires augmente avec la taille du réseau, ce qui justifie a posteriori le choix de représentation fait en (Hervé et al. (90a)). Dans les Figures 67-92, on observe, comme nous l'avons remarqué plus haut pour les histogrammes d'apparition des singletons, paires, ..., que le mode parallèle privilégie les cliques et amas de cardinal faible ou de cardinal grand, alors que le mode bloc-séquentiel privilégie les cliques et amas de cardinal moyen, ce que l'on peut résumer ainsi : le mode parallèle montre plus de neurones en activité, mais moins d'interactions de paires. Si l'on est dans le cas simple de la p. 40, dans lequel les poids n'évoluent pas et dans lequel la matrice W est symétrique, alors, un simple calcul montre que le potentiel de singletons s'écrit :

- dans le cas bloc-séquentiel choisi ici (à éléments d'un bloc non voisins) :

$$J_U^S (\{i\}) = \log (e^{W_{ii}})$$

- dans le cas parallèle :

$$J_U^P (\{i\}) = \log (1 + e^{W_{ii}}),$$

à condition de choisir dans les deux cas $J_U^P (\emptyset) = J_U^S (\emptyset) = 0$.

Ceci nous prouve que, dans ce cas simple, le mode parallèle présente bien une plus grande activité des neurones que le mode bloc-séquentiel, mais une moins grande interaction de paires.

Dans les cas les plus complexes (évolution hebbienne des poids, W non symétrique), les simulations numériques ont montré le même phénomène (cf. Figures 67-92 et Annexe 2 ; cf. aussi Hervé et al. (90b), dans lequel une diminution de synchronisme, entraînée par une diminution de la période réfractaire, diminue l'activité, mais renforce les interactions de paires) ; il serait donc très intéressant maintenant d'essayer de prouver théoriquement ce résultat dans les réseaux complexes à poids variables non symétriques. Une approche thermodynamique (François (90)) permettrait sans doute de progresser dans cette direction.

ANNEXE 1

*Méthode d'estimation, test du Khi-deux
et Programmes*

PARTIE (A)

Méthode d'estimation et test du Khi-deux

1° - Méthode d'estimation fonctionnelle : Méthode du Noyau

Soient X_0, \dots, X_p , *i.i.d* de loi f , $(p+1)$ modalités, n_0, \dots, n_p , $(p+1)$ effectifs observés et $n = \sum_{i=0}^p n_i$; $i = 0, \dots, p$, la taille totale de l'échantillon.

D'après un théorème (*cf. A. Berlinet (86)*), après avoir vérifié les conditions imposées il existe un noyau K_E et un pas H_0 optimaux, tels que : l'estimateur f_n converge vers f où

$$f_n(x) = \frac{1}{n H_0} \sum_{i=0}^p n_i K_E\left(\frac{x - x_i}{H_0}\right)$$

$$\text{avec } H_0 = \left[\frac{\int K_E(x) dx}{\left[\int x^2 K_E(x) dx \right]^2 \int [f''(x)]^2 dx} \right]^{\frac{1}{5}} \cdot n^{-\frac{1}{5}}$$

et où K_E est appelé noyau d'Epanéchnikov et f'' est la dérivée seconde de la densité f .

Définition : On définit $eff(K)$, l'efficacité d'un noyau K par rapport à K_E (noyau d'Epanéchnikov), par :

$$eff(K) = \left[\frac{C(K_E)}{C(K)} \right], \text{ où}$$

$$C(K) = \left[\int x^2 K(x) dx \right]^{\frac{2}{5}} \left[\int (K(x))^2 dx \right]^{\frac{4}{5}}$$

Dans la pratique, on a pris le noyau K (cf. Figure 1) tel que son efficacité par rapport au noyau optimal K_E est : $eff(K) = 1.001$ asymptotiquement ; il est défini par :

$$K(x) = \begin{cases} \frac{\sqrt{\pi^2 - 8}}{4} \cos \left[\frac{\sqrt{\pi^2 - 8}}{2} x \right], & \text{si } |x| \leq \frac{\pi}{\sqrt{\pi^2 - 8}} \\ 0 & , \text{ sinon} \end{cases}$$

$$\text{et } \int x^2 K(x) dx = 1, \quad \int (K(x))^2 dx = \frac{\pi \sqrt{\pi^2 - 8}}{16}$$

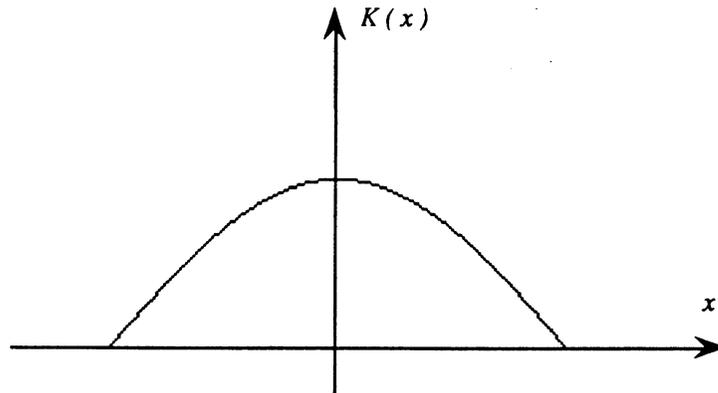


Figure 1 : Densité du noyau "K"

* **Calcul pratique du pas "H₀" :**

Le calcul est fait en deux étapes :

1° - calculons la première estimation $f_{n1}(x)$ comme suit :

$$f_{n1}(x) = \frac{1}{n H_1} \sum_{i=0}^p n_i K\left(\frac{x - x_i}{H_1}\right) \text{ où } H_1 = \frac{S}{n^{1/5}}$$

et où S^2 est la variance empirique de l'échantillon.

2° - remplaçons f par f_{nl} dans la formule du H_0 donnée ci-dessus.

Après un long calcul, on a trouvé la formule explicite de H_0 comme suit :

$$H_0 = \left[\frac{16 \pi n (H_1)^5}{A + b \sum_{i=0}^p n_i^2} \right]^{\frac{1}{5}} \cdot \frac{b}{\pi}$$

$$\text{où } A = \sum_{i=0}^{p-1} \sum_{j=i+1}^p n_i n_j \int_{\left\{ |x| \leq b \right\} \cap \left\{ \left| x + \frac{x_i - x_j}{H_1} \right| \leq b \right\}} \left[\cos \left(2ax + a \frac{x_i - x_j}{H_1} \right) + \cos \left(a \frac{x_i - x_j}{H_1} x \right) \right] dx$$

$$\text{et où } b = \frac{\pi}{\sqrt{\pi^2 - 8}}, a = \frac{\sqrt{\pi^2 - 8}}{2}$$

Notamment, on a implémenté cette méthode sur le Vax6000 (cf. annexe 1-partie (B)).

2° - Lois usuelles et Test du Khi-deux :

* Lois usuelles utilisées :

1 - Loi Normale : $\mathcal{N}(m, \sigma)$

$$f_{\mathcal{N}(m, \sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[x-m]^2}{2\sigma^2}} \mathbb{1}_{\mathbb{R}}(x)$$

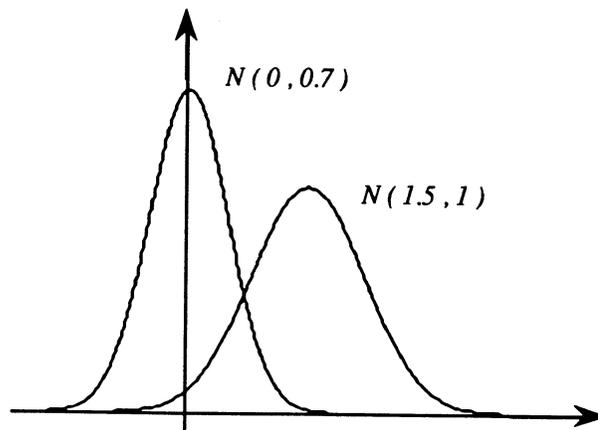


Figure 2 : Densité de la loi Normale : $\mathcal{N}(m, \sigma)$

2 - Loi Log-Normale : $LN(m, \sigma)$

$$f_{LN(m, \sigma)}(x) = \frac{1}{\sqrt{2\pi} \sigma x} e^{-\frac{[\log(x) - m]^2}{2\sigma^2}} \mathbb{1}_{\mathbb{R}_+^*}(x)$$

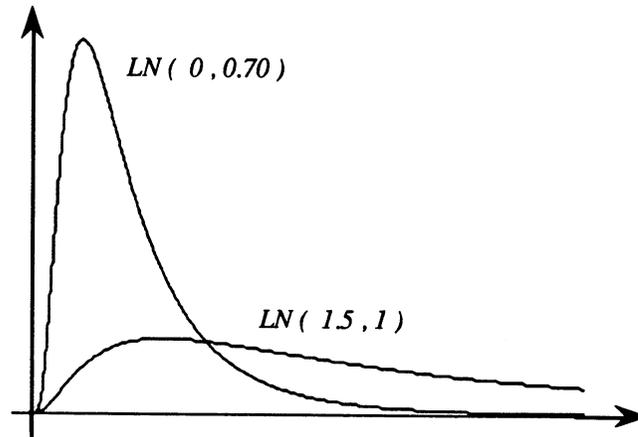


Figure 3 : Densité de la loi Log-Normale : $LN(m, \sigma)$

3 - Loi Gamma : $\Gamma(a, b)$

$$f_{\Gamma(a, b)}(x) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}} \mathbb{1}_{\mathbb{R}_+}(x)$$

$$\text{avec } \Gamma(a) = \int_0^{+\infty} x^{a-1} e^{-x} dx$$

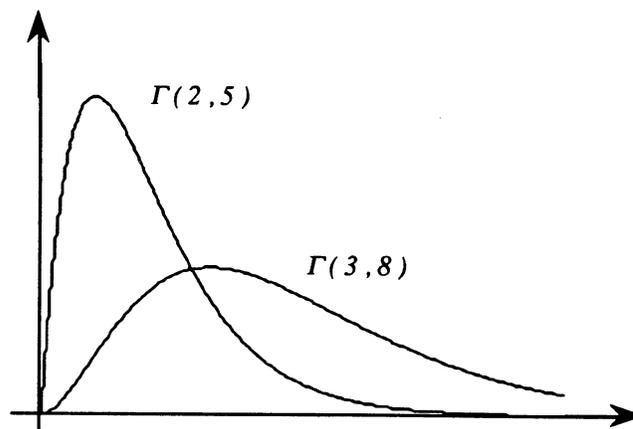


Figure 4 : Densité de la loi Gamma : $\Gamma(a, b)$

4 - Loi Log-Gamma : $L\Gamma(a)$

$$f_{L\Gamma(a)}(x) = \frac{1}{\Gamma(a)} e^{(ax - e^x)} \mathbb{1}_{\mathbb{R}}(x)$$

$$\text{avec } \Gamma(a) = \int_0^{+\infty} x^{a-1} e^{-x} dx$$

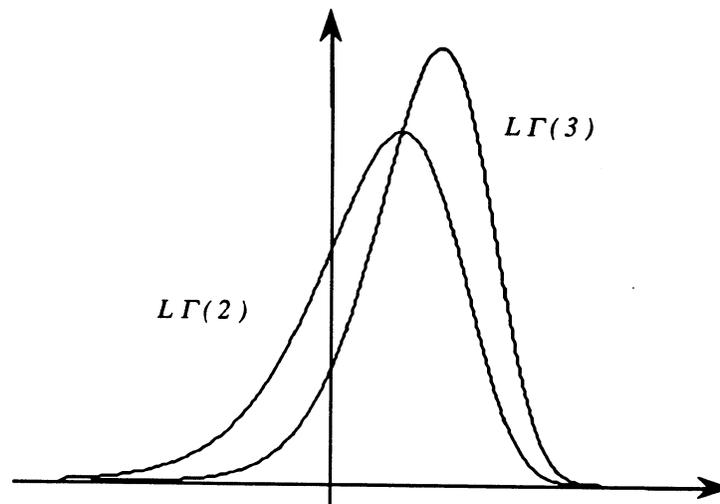


Figure 5 : Densité de la loi Log-Gamma : $L\Gamma(a)$

5 - Loi Khi-deux à n d° de libertés: $\chi^2(n) = \Gamma(n/2, 2)$

$$f_{\chi^2(n)}(x) = \frac{1}{2^{n/2} \Gamma(\frac{n}{2})} x^{(n/2)-1} e^{-\frac{x}{2}} \mathbb{1}_{\mathbb{R}_+}(x)$$

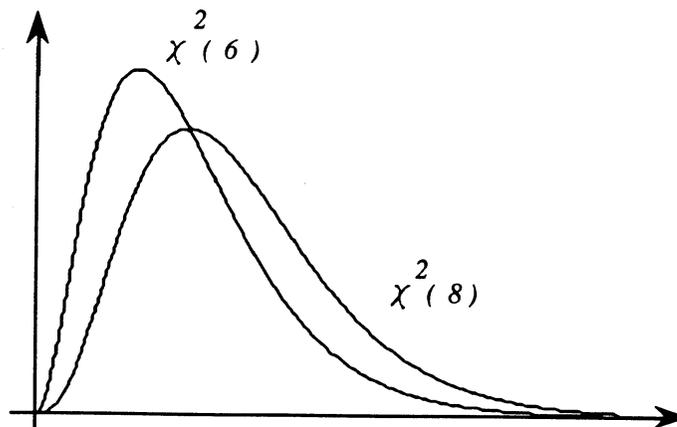


Figure 6 : Densité de la loi du Khi-deux $\chi^2(n)$

6 - Loi de Weibull : $\mathcal{W}eib(m, a, c)$

$$f_{\mathcal{W}eib(m, a, c)}(x) = \frac{c}{a} \left[\frac{x-m}{a} \right]^{c-1} e^{-\left[\frac{x-m}{a} \right]^c} \mathbb{1}_{\mathbb{R}_+}(x-m)$$

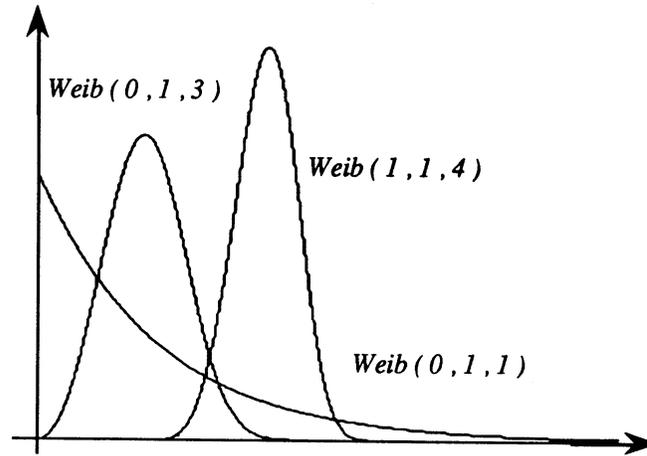


Figure 7 : Densité de la loi de Weibull : $\mathcal{W}eib(m, a, c)$

7 - Loi de Poisson : $\mathcal{P}(\lambda)$

$$\forall n \in \mathbb{N}, \mathcal{P}(\lambda)[n] = \frac{\lambda^n}{n!} e^{-\lambda}$$

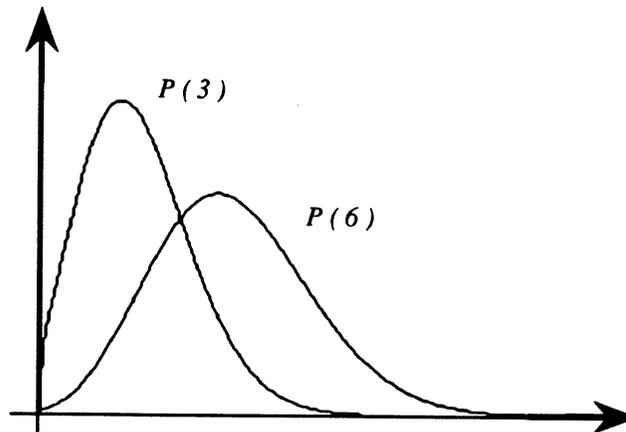


Figure 8 : Densité de la loi de Poisson : $\mathcal{P}(\lambda)$

8 - Loi Binomiale : $\mathcal{B}(N, p)$

$$\forall N \in \mathbb{N}^*, \forall n \in \{0, \dots, N\} \forall P \in [0, 1]$$

$$\mathcal{B}(N, P)[n] = C_N^n P^n (1 - P)^{N-n}$$

$$\text{où } C_N^n = \frac{N!}{(N-n)! n!}$$

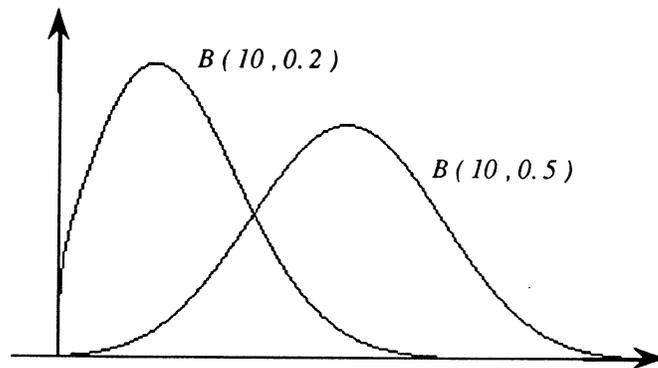


Figure 9 : Densité de la loi Binomiale : $\mathcal{B}(N, p)$

* Test du Khi-deux :

Soient X_0, \dots, X_p , *i.i.d* de loi f , $(p+1)$ modalités, n_0, \dots, n_p , $(p+1)$ effectifs observés et $n = \sum_{i=0, \dots, p} n_i$, la taille totale de l'échantillon. On calcule les fréquences théoriques (P_i) ; $i = 0, \dots, p$, de la loi par rapport à laquelle on teste notre échantillon ; ses paramètres sont estimés à partir de l'échantillon en question, tels que :

$$- P_i = \text{Prob}(\{x_i\}) \quad (\text{cas d'une loi discrète})$$

$$- P_i = F(x_i + \varepsilon) - F(x_i - \varepsilon) \quad (\text{cas d'une loi continue})$$

où F est la fonction de répartition et où 2ε est la taille de la fenêtre de l'histogramme des (X_i) ; $i = 0, \dots, p$.

La statistique T observée du Khi-deux est donnée par :

$$T = \sum_{i=0}^p \frac{(n \cdot p_i - n_i)^2}{n \cdot p_i}$$

En vertu d'un résultat probabiliste, la loi de la statistique T , quand n , la taille de l'échantillon, est assez grand, suit une loi du $\chi^2[(p+1) - 1 - nb]$, où nb est le nombre de paramètres estimés.

Enfin, le test du Khi-deux consiste à tester l'hypothèse H_0 : "la loi de l'échantillon est la loi théorique" contre l'alternative H_1 : "elle ne l'est pas". Donc, on rejette l'hypothèse " H_0 " dès que la valeur de la statistique observée T a franchi un certain seuil A_α , donné selon un risque α (appelé risque du premier espèce, donné en unité de pourcentage), sinon on accepte " H_1 ", c'est-à-dire que l'on détermine A_α par :

$$\text{Prob} \{ \chi^2[(p+1) - 1 - nb] \geq A_\alpha \} = \alpha \quad \text{alors}$$

- * Si $T \geq A_\alpha$, on rejette l'hypothèse " H_0 ",
- * Sinon, on accepte l'hypothèse " H_0 " (cf. Figure 10).

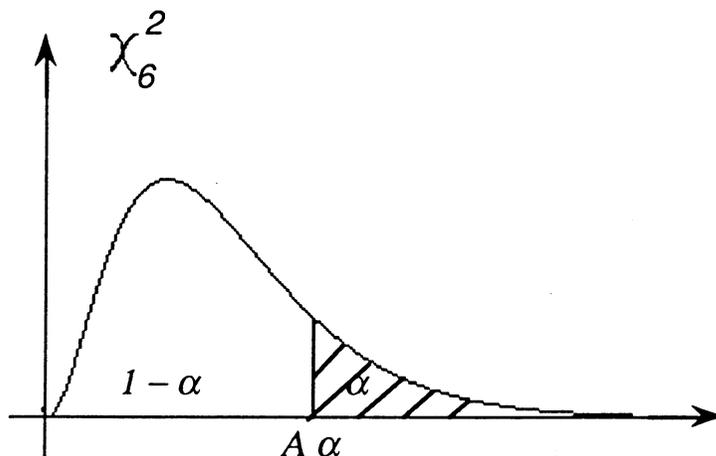


Figure 10 : Régions d'acceptation et de rejet

Remarque : Cette annexe a été réalisée en consultant plusieurs ouvrages et revues (cf. Jagdish & al. (76), Berlinet(86), Lawless(82), Johnson & al. (70), Khelladi & al. (87)).

PARTIE (B)

Programmes en langage "C"


```

scanf("\n%is", &f_out2);
printf("%s\n", f_out2);
if ( (fichout2=fopen(f_out2, "w"))==0 )
    (printf(" Erreur d'écriture : \n");)
)
)
/*-----*/
/* Procedure : proc_lecture() */
/*-----*/
proc_lecture()
{
    int i;
    N=0;
    i=0;
    while ( i < Np )
        (
            fscanf(fichin, "%d %d", &G[i], &T[i]);
            /*printf("T[%d] = %d\n", G[i], T[i]);*/
            N+=T[i];
            i++;
        )
    pp=i-1;
    /*printf("pp = %d\n", pp, N);*/
    fclose(fichin);
}
/*-----*/
/* Procedure : proc_écriture() */
/*-----*/
proc_écriture(ptr_out, T1, T2, n)
FILE *ptr_out;
double T1[101], T2[101];
int n;
{
    int i;
    i=0;
    while ( i<n )
        (
            fprintf(ptr_out, "%2f %4f", T1[i], T2[i]);
            fprintf(ptr_out, "\n");
            /*printf("T[%2f] = %e\n", T1[i], T2[i]);*/
            i++;
        )
    printf("\n");
    fclose(ptr_out);
}
/*-----*/
/* Fonction : puissance() */
/*-----*/
double puissance(a, k)
int k;
double a;
{
    int i;
    double puis;
    puis=1;
    for (i=1; i<=k; i++)
        (
            puis*=a;
        )
    return (puis);
}
)
)
/*-----*/
/* Procedure : init_parametres() */
/*-----*/
init_parametres()
{
    int i;
    double som, mgeo;
    double som2, som3, som4;
    double skew, kurt, tamp;
    mgeom=1;
    som=0.0; som2=0.0;
    som3=0.0; som4=0.0;
    N2=0;
    for (i=0; i<=pp; i++)
        (
            N2+= T[i];
            som += G[i]*T[i];
            som2 += G[i]*G[i]*T[i];
            som3 += puissance((double)G[i], 3)*T[i];
            som4 += puissance((double)G[i], 4)*T[i];
            mgeom *= exp(log((double)G[i]))*((double)T[i]/N);
        )
    /*mgeo=exp(log(fabs(mgeo))/N);*/
    moy=som/N;
    som2/=N;
    som3/=N;
    som4/=N;
    sigma=sqrt(som2-moy*moy);
    /*printf("sigma = %4f\n", sigma);*/
    sigma=sqrt(((double)N)/((double)N+1))*sigma;
    printf("\n");
    printf(" * moyenne arithmétique = %4f\n", moy);
    printf(" * moyenne géométrique = %4f\n", mgeo);
    printf(" * SQRT(variance empirique) = %4f\n", sigma);
    printf("\n");
    if ( choixproc=='n' )
        (
            tamp=puissance(moy, 3);
            skew=som3-3*som2*moy+2*tamp;
            tamp=puissance(sigma, 3);
            skew/=tamp;
            tamp=puissance(moy, 4);
            kurt=som4-4*som3*moy+6*som2*moy*moy-3*tamp;
            tamp=puissance(sigma, 4);
            kurt/=tamp;
            kurt-=3;
        )
    printf(" * Test de normalité : \n\n");
    printf(" Si X une v.a. suivant une loi normale alors : \n");
    printf(" - Le Skewness=E((X-EX)**3)/sigma**3 = 0 (symétrique, \n");
    printf(" - Le Kurtosis=E((X-EX)**4)/sigma**4-3 = 0 (forme gauss), \n");
    printf("\n\n");
    printf(" | loi gauss | Skewness | Kurtosis | \n");
    printf(" |-----|-----|-----| \n");
    printf(" | theorique | 0 | 0 | \n");
    printf(" | calculé | %6.3f | %6.3f | \n", skew, kurt);
    printf(" |-----|-----|-----| \n");
}
)
)

```

```

if( choixproc=='b')
  ( moy/'pp; )
)
/*-----*/
/* Fonction : factoriel() */
int factoriel(s)
int s;
{
  int i, res;
  res=1;
  for(i=s;i>0;i--)
  {
    res*=i;
  }
  return(res);
}
/*-----*/
/* Fonction : poisson() */
double poisson(e,n)
double e;
int n;
{
  double res;
  /*printf("moy= %e\n", moy);*/
  res=puissance(e,n);
  res*=exp(-e);
  res/=factoriel(n);
  return(res);
}
/*-----*/
/* Fonction : combinatoire() */
double combinatoire(i,m)
int i,m;
{
  double res;
  int d, nu, j;
  nu=i;
  j=0;
  while(j<i)
  {
    nu*=(m-j);
    j++;
  }
  d=factoriel(i);
  res=(double)nu/(double)d;
  return(res);
}
/*-----*/
/* Fonction : binomiale() */
double binomiale(i,m,p)
int i,m;
double p;
{
  double res, moy2;
  int j;
  moy2=1-p; j=m-i;
}
res=puissance(p,i);
res*=puissance(moy2,j);
res*=combinatoire(i,m);
return(res);
}
/*-----*/
/* Fonction : MMoy() */
/* cette fonction calcule la moyenne empirique d'un echantillon */
double MMoy(T1,n)
double T1[201];
int n;
{
  double s;
  int i;
  s=0;
  for(i=0;i<=n;i++)
  {
    s+=T1[i];
  }
  s/=n;
  return(s);
}
/*-----*/
/* Fonction : Cov() */
/* cette fonction calcule la covariance de T1 et T2 */
double Cov(T1,T2,n)
double T1[201], T2[201];
int n;
{
  double res, sxy;
  int i;
  sxy=0;
  for(i=0;i<=n;i++)
  {
    sxy+=T1[i]*T2[i];
  }
  res=sxy/n-MMoy(T1,n)*MMoy(T2,n);
  return(res);
}
/*-----*/
/* Fonction : fct() */
double fct(x)
double x;
{
  double res;
  res=log(log(1./(1. - x)));
  return(res);
}
/*-----*/
/* Procure : Estim_m() */
/* cette procedure calcule l'estimation de location "m" d'une */
/* loi de Weibull(m,a,c) par le Max-vraisemblance. */
Estim_m()
{
  int i;
  if(param=='o')
  {
    nbpar= 3;
    vm=(double)G[0]; /* minimum des observations G[i] */
  }
  else
}

```

```

( nbpar= 2;
  wm=0.0;
)
for(i=1;i<=pp;i++)
{
  xz[i]=log((double)G[i]-wm);
  YY[i]=fct((double)i/(pp+1));
  xz[0]=xz[i];
  YY[0]=YY[i];
}
/*-----*/
/*
  Procedure : Estim_Graph()
  cette procedure calcule l'estimation de "a","c" d'une loi
  de Weibull(m,a,c) par l'ajustement de la droite de Henry.
  */
/*-----*/
Estim_Graph(T1,T2,n)
double T1[201],T2[201];
int n;
{
  double res,coefcorr;
  cgr=Cov(T1,T2,n)/Cov(T1,T1,n);
  res=MMoy(T2,n)-cgr*MMoy(T1,n);
  coefcorr=Cov(T1,T2,n)/sqrt(Cov(T1,T1,n)*Cov(T2,T2,n));
  agr=exp(-res/cgr);
  /*printf("Les Parametres estimes par la droite de Henry sont : \n\n");
  printf("wm = %e agr = %e cgr = %e\n\n",wm,agr,cgr);
  printf("le coefficient de correlation : RO = %e\n",coefcorr);*/
}
/*-----*/
/*
  Fonction : fct_Phy()
  */
/*-----*/
double fct_Phy(T1,T2,p)
double T1[201],T2[201];
int p;
{
  double d,e,f,g,h,u,phi,phiprime,res;
  int i;
  d=0;e=0;f=0;g=0;
  for(i=0;i<=p;i++)
  {
    h=log(T1[i]);
    u=exp(h*wc);
    d+=u*h*T2[i];
    e+=u*T2[i];
    f+=h*T2[i];
    g+=u*h*h*T2[i];
  }
  phi=1./wc + f/N - d/e;
  phiprime= -1./(wc*wc) - g/e + (d/e)*(d/e);
  res = wc - phi/phiprime;
  return(res);
}
/*-----*/
/*
  Fonction : Estim_a()
  */
/*-----*/
double Estim_a(T1,T2,p)
double T1[201],T2[201];
int p;
{
  double s,res;
  int i;
  s=0;
  for(i=0;i<=p;i++)
  {
    s+=T2[i]*exp(wc*log(T1[i]));
  }
  res=exp(log(s/N)/wc);
  return(res);
}
/*-----*/
/*
  Procedure : Newton()
  cette procedure calcule l'estimation de "a","c" d'une loi
  de Weibull(m,a,c) par methode iterative de Newton-Raphson.
  */
/*-----*/
Newton(T1,T2,p)
double T1[201],T2[201];
int p;
{
  double eps,cl;
  int iter;
  wa=agr;
  wc=cgr;
  eps=1;
  iter=0;
  /*printf("la valeur initiale de wc est : %6f\n",wc);*/
  while(eps>1.0E-06)
  {
    cl=wc;
    wc=fct_Phy(T1,T2,p);
    eps=fabs(wc-cl);
    iter++;
    /*printf("Iteration Numero:%d\twc = %6f\n",iter,wc);*/
  }
  wa=Estim_a(T1,T2,p);
  /*printf("Les parametres estimes par Max-vraisemblance sont : \n\n");
  printf("wm = %e wa = %e wc = %e\n\n",wm,wa,wc);*/
}
/*-----*/
/*
  Fonction : Frweibull()
  cette fonction calcule la fonction de répartition
  de la loi de Weibull(wm,wa,wc).
  */
/*-----*/
double Frweibull(x)
double x;
{
  double res;
  if(x>wm)
  {
    res=exp(wc*log((x-wm)/wa));
    res=1.0 - exp(-res);
  }
  else
  {
    res=0.0;
  }
  return(res);
}
/*-----*/
/*
  Fonction : Frgauss_normale()
  cette fonction calcule la fonction de répartition
  de la loi normale centrée et réduite.
  */
/*-----*/
double Frgauss_normale(x)
double x;
{
  double res;
}

```



```

/*-----*/
/*
/* Fonction : densite_gamma(x)
/* cette fonction calcule: la densite de la loi Gamma (ga,gb)
/*-----*/
double densite_gamma(x,Gaga,nl)
double x;
double Gaga[20];
{
    int i;
    double res,tampon;
    if(x==0)
        [res=0;]
    else
    {
        res=exp((-x/gb)+(ga-1)*log(fabs(x)));
        tampon=exp(ga*log(fabs(gb)));
        res/=tampon;
        for(i=0;i<nl;i++)
            [res/=Gaga[i];]
    }
    return(res);
}
/*-----*/
/*
/* Fonction : Frgamma()
/* cette fonction calcule: Fr(b)-Fr(a)=Prob(x appartient [a,b])
/*-----*/
double Frgamma(a,b,Gaga,nl)
double a,b;
double Gaga[20];
{
    double res,tampon,abs,ord,delta2;
    int i;
    res=0.0;
    tampon=densite_gamma(a,Gaga,nl);
    delta2=(b-a)/100;
    i=1;
    while(i<=100)
    {
        abs = a + i*delta2;
        ord = densite_gamma(abs,Gaga,nl);
        res += delta2*(tampon + ord)/2;
        tampon = ord;
        i++;
    }
    return(res);
}
/*-----*/
/*
/* Estimation d'une loi de densite par la methode du noyau.
/* (cf. BENAOUA et al.,rapport de D.E.A. Statistique (1987)).
/*-----*/
/*
/* Procedure : calcul_h0()
/*-----*/
{
    int i,j;
    double a,b;
    double borne,In,I1,I2;

```

```

/*printf("la valeur initiale de ga est : %.6f\n",ga);*/
while(eps>1.0E-06)
{
    cl=ga;
    ga=fnct_Phy_gamma();
    eps=fabs(ga-cl);
    iter++;
    /*printf("Iteration Numero:%d\tga = %.6f\n",iter,ga);*/
}
gb=moy/ga;
/*printf("Les parametres estimes par Max-vrais (Newton) sont :\n\n");
printf("ga = %e gb = %e\n",ga,gb);*/
}
/*-----*/
/*
/* Fonction : gamma()
/* calcule l'integrale d'Euler.
/*-----*/
double gamma(x)
double x;
{
    double res,y,c1,c2,c3,c4;
    double c5,c6,c7,c8;
    c1=0.0016063118;
    c2=0.0051589951;
    c3=0.0044511400;
    c4=0.0721101567;
    c5=0.0821117404;
    c6=0.4117741955;
    c7=0.4227874605;
    c8=0.9999999758;
    res=1;
    y=x;
    if(y==0)
    {
        res = 1.0E+18;
        goto one;
    }
    two : if(y<2.0)
    {
        res/=y;
        y+=1;
        goto two;
    }
    three : if(y==2.0)
    {
        goto one;
    }
    if(y>3.0)
    {
        y-=1;
        res*=y;
        goto three;
    }
    y-=2;
    res=((((c1*y+c2)*y+c3)*y+c4)*y+c5)*y+c6)*y+c7)*y+c8)*res;
    one :
    return(res);
}

```

```

double test,h1,nu,d;
h1=sigma/exp(log((double)N)/5);
a=sqrt(pi*pi-8)/2;
b=pi/(2*a);
borne=2*b*h1;
In=0.0;
for(i=0;i<=pp-1;i++)
  for(j=i+1;j<=pp;j++)
    test=(double)(G[i]-G[j]);
    if(test>0.0)
      if(test < borne )
        {
          I1=cos(a*(test/h1))*(2*b+test/h1);
          I2=sin(a*(2*b+test/h1))/a;
          In+=(I1+I2)*T[i]*T[j];
        }
      else
        if(test > -borne )
          {
            I1=cos(a*(test/h1))*(2*b+test/h1);
            I2=sin(a*(2*b+test/h1))/a;
            In+=(I1+I2)*T[i]*T[j];
          }
    }
nu=16*pi*N*puissance(h1,5);
d=N2*b*In;
h0=exp(log(fabs(nu/d))/5)*b;
h0/=pi;
}
/*-----*/
/* Fonction : noyau()
double noyau(x)
double x;
{
  double res,s,s2;
  s =pi/sqrt(pi*pi-8);
  s2=sqrt(pi*pi-8)/4;
  if( (x<=s) && (x>= -s) )
    ( res=s2*cos(2*s2*x);)
  else
    ( res =0.0; )
  return(res);
}
/*-----*/
/* Fonction : densite empir()
double densite_empir(x,h)
double x,h;
{
  double res;
  int i;
  res=0.0;
  for(i=0;i<=pp;i++)
    ( res+=(double)T[i]*noyau((x-(double)G[i])/h);)
  res/= (N*h);
  return(res);
}
/*-----*/
/* Procedure : density estimates()
cette procedure calcule la densite empirique a noyau
d'un echantillon.
density_estimates()
{
  int i;
  double sum;
  sum=0.0;
  calcul_h0();
  printf(" * Le pas optimal de la densite a noyau: HO = %.4f\n",h0);
  h0*=coef; * Le pas utilise : HO = HO*coef (optimal) = %.4f\n",h0);
  printf("
  for(i=0;i<=pp;i++)
    {
      Tx[i] = (double)G[i];
      Tnoy[i] = densite_empir(Tx[i],h0);
      sum+=Tnoy[i];
      /*printf("Tx[%4d]=%.4f Ty[%4d]=%.4f\n",i,Tx[i],i,Tnoy[i]);*/
    }
  printf(" * la some des Tnoy[i] :
  printf("\n");
}
/*-----*/
/* Fonction : arrondi(x)
cette procedure calcule l'entier le plus proche du
real x
arrondi(x)
double x;
{
  int res;
  double test;
  res=(int)x + 1;
  test=(double)res-x;
  if(test<=0.5)
    ( return(res); )
  else
    ( return(res-1); )
}
/*-----*/
/* Procedure : proc_khi_deux()
proc_khi_deux(nb)
int nb;
{
  int i;
  spi=0.0;
  khideux=0.0;

```

```

double test,h1,nu,d;
h1=sigma/exp(log((double)N)/5);
a=sqrt(pi*pi-8)/2;
b=pi/(2*a);
borne=2*b*h1;
In=0.0;
for(i=0;i<=pp-1;i++)
  for(j=i+1;j<=pp;j++)
    test=(double)(G[i]-G[j]);
    if(test>0.0)
      if(test < borne )
        {
          I1=cos(a*(test/h1))*(2*b+test/h1);
          I2=sin(a*(2*b+test/h1))/a;
          In+=(I1+I2)*T[i]*T[j];
        }
      else
        if(test > -borne )
          {
            I1=cos(a*(test/h1))*(2*b+test/h1);
            I2=sin(a*(2*b+test/h1))/a;
            In+=(I1+I2)*T[i]*T[j];
          }
    }
nu=16*pi*N*puissance(h1,5);
d=N2*b*In;
h0=exp(log(fabs(nu/d))/5)*b;
h0/=pi;
}
/*-----*/
/* Fonction : noyau()
double noyau(x)
double x;
{
  double res,s,s2;
  s =pi/sqrt(pi*pi-8);
  s2=sqrt(pi*pi-8)/4;
  if( (x<=s) && (x>= -s) )
    ( res=s2*cos(2*s2*x);)
  else
    ( res =0.0; )
  return(res);
}
/*-----*/
/* Fonction : densite empir()
double densite_empir(x,h)
double x,h;
{
  double res;
  int i;
  res=0.0;
  for(i=0;i<=pp;i++)
    ( res+=(double)T[i]*noyau((x-(double)G[i])/h);)
  res/= (N*h);
  return(res);
}
/*-----*/
/* Procedure : proc_khi_deux()
proc_khi_deux(nb)
int nb;
{
  int i;
  spi=0.0;
  khideux=0.0;

```

```

)
i++;
k=i;
while (k<=j1-1)
(
  if (pos[k]<=j)
    { i++; k++; }
)
k=cptr+1;
while (k<=t-1)
(
  Tint[k] = T[k];
  nTint[k] = nT[k];
  k++;
)
)
else
(
  for (i=0; i<=t-1; i++)
    { Tint[i] = T[i];
      nTint[i] = nT[i];
    }
)
)
/*-----*/
/* Procedure : proc traitement2()
/*-----*/
proc traitement2(t)
inE t;
(
  double s,sn;
  int pos[101],j1,i,k,j,cptr;
  j1=0;
  for (i=t; i<=pp; i++)
    { if (nT[i]<5)
      (
        pos[j1]=i;
        j1++;
      )
    }
  /*printf("j1 = %d\n",j1);*/
  if (j1!=0)
    { i=pos[j1-1]+1;
      while (i<=pp)
        { Tint[i] = T[i];
          nTint[i] = nT[i];
          i++;
        }
      i=j1-1;
      while (i>=0)
        { sn = T[pos[i]];
          s = nT[pos[i]];
          j=pos[i]-1;
          while (j>=t)

```

```

  degree=0;
  for (i=0; i<=pp; i++)
    { spi=nT[i];
      if (nTint[i] >= 5 )
        { degree++;
          kh2[i]=((double)Tint[i]-nTint[i])*((double)Tint[i]-nTint[i])/nTint[i];
          khideux += kh2[i];
        }
    }
  spi/=N;
  spi=arrondi(spi);
  degree -= (nb+1);
)
/*-----*/
/* Procedure : proc traitement1()
/*-----*/
proc traitement1(t)
inE t;
(
  double s,sn;
  int pos[101],j1,i,k,j,cptr;
  j1=0;
  for (i=0; i<=t-1; i++)
    { if (nT[i]<5)
      (
        pos[j1]=i;
        j1++;
      )
    }
  /*printf("j1 = %d\n",j1);*/
  if (j1!=0)
    { i=0;
      while (i<=pos[0])
        { Tint[i] = T[i];
          nTint[i] = nT[i];
          i++;
        }
      i=0;
      while (i<=j1-1)
        { sn = T[pos[i]];
          s = nT[pos[i]];
          j=pos[i]+1;
          while (j<=t-1)
            { s += nT[j];
              sn += T[j];
              if (s>=5)
                { Q[j] = sn;
                  nQ[j] = s;
                  Tint[j]=Q[j];
                  nTint[j]=nQ[j];
                  cptr = j;
                  break;
                }
            }
          j++;
        }
    }

```



```

imprime_tab3();
if(choix1=='o')
(
/*printf("---- densite a noyau----\n");*/
proc_écriture(fichout1,Tx,Tnoy,pp);
)
if(choix2=='o')
(
/*printf("---- densite theorique ----\n");*/
proc_écriture(fichout2,Tx,Tth,pp);
break;
)
) /* fin switch(choixproc) */
) /* fin main() */

imprime_tab3();

case 'p' :
(
nbpar = 1;
i=0;
Tth[i]= poisson(moy,G[i]);
nT[i]=N*Tth[i];
i=1;
while(i<=pp)
(
Tth[i]= (moy/G[i])*Tth[i-1];
nT[i]=N*Tth[i];
i++;
)
density_estimates();
proc_traitement1(p1);
proc_traitement2(p1);
proc_khi_deux(nbpar);
printf("\n\n");
printf(" * la loi théorique est la loi Poisson( %.2f )",moy);
imprime_tab3();
if(choix1=='o')
(proc_écriture(fichout1,Tx,Tnoy,pp);)
if(choix2=='o')
(proc_écriture(fichout2,Tx,Tth,pp);)
break;
)
case 'b' :
(
nbpar = 1;
i=0;
Tth[i]= binomiale(G[i],pp,moy);
nT[i]=N*Tth[i];
i=1;
while(i<=pp)
(
Tth[i]=((pp-G[i-1])*moy)/(G[i]*(1-moy))*Tth[i-1];
nT[i]=N*Tth[i];
i++;
)
density_estimates();
proc_traitement1(p1);
proc_traitement2(p1);
proc_khi_deux(nbpar);
printf("\n\n");
printf(" * la loi théorique est la loi Binomiale(%d, %.2f)",pp,moy);

```

ANNEXE 2

***Visualisations, schémas des exécutions
et programmes***

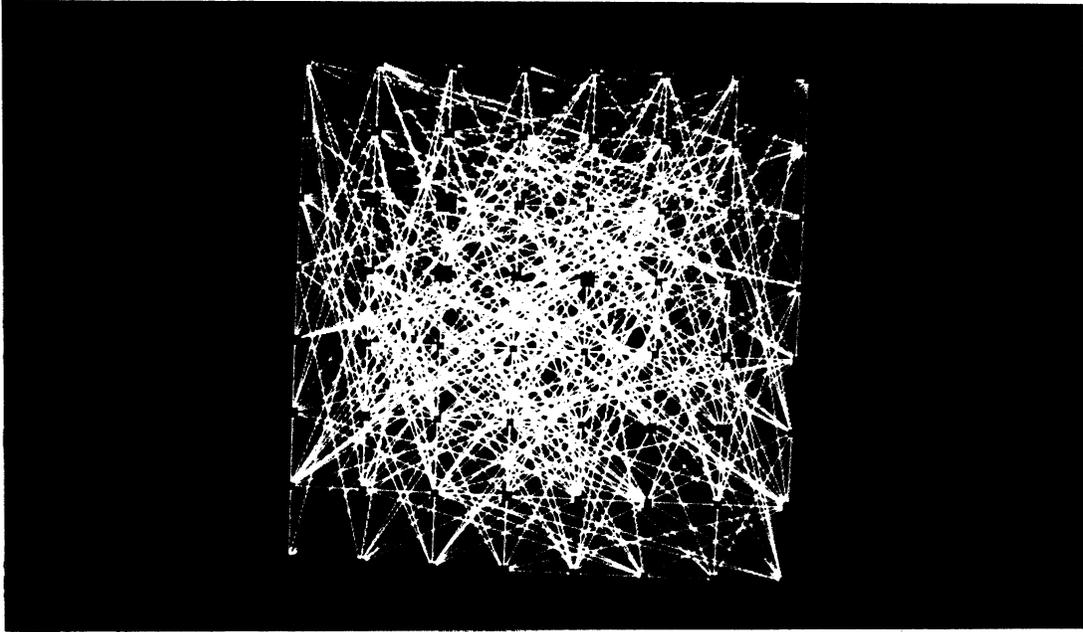
PARTIE (A)**Visualisations des potentiels d'interactions,
Cliques et Amas des réseaux en couleurs**

Figure 1 : Réseau de neurones de taille $N = 8 \times 8$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. §IV -3.2.2, Figure 58). Vu qu'il y a plusieurs niveaux de potentiels d'interaction de singletons, on les a regroupés par bloc et, ensuite, on les a représentés par des carrés rouges dont la dimension est proportionnelle à la valeur du potentiel commun. Représentation des potentiels d'interaction de paires par des barres vertes.

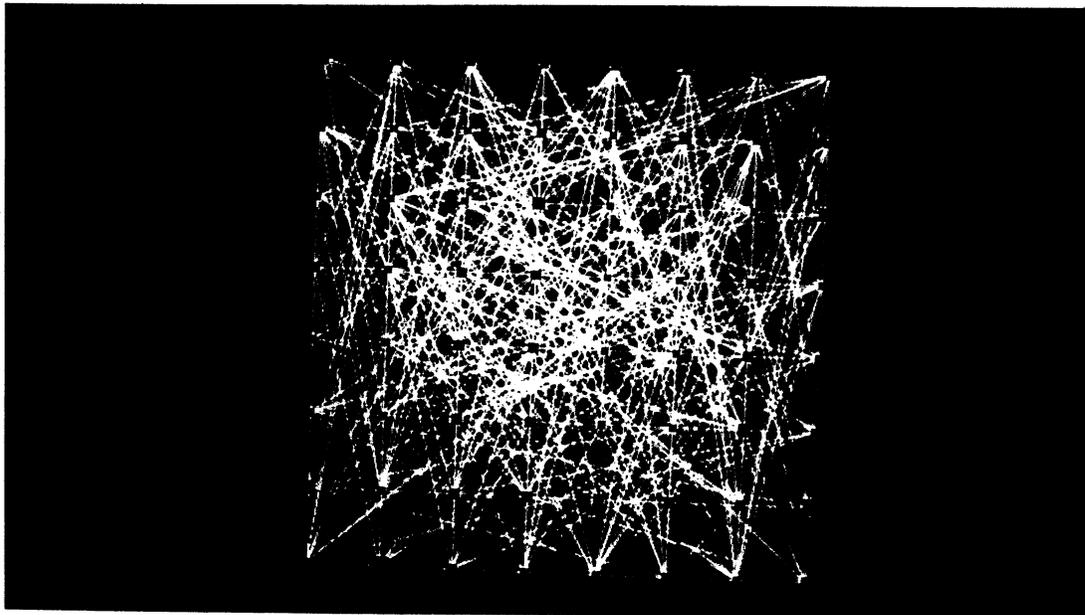


Figure 2 : Réseau de neurones de taille $N = 8 \times 8$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. §IV -3.2.2, Figure 58). Vu qu'il y a plusieurs niveaux de potentiels d'interaction de singletons, on les a regroupés par bloc et, ensuite, on les a représentés par des carrés rouges dont la dimension est proportionnelle à la valeur du potentiel commun. Représentation des potentiels d'interaction de paires par des barres vertes.

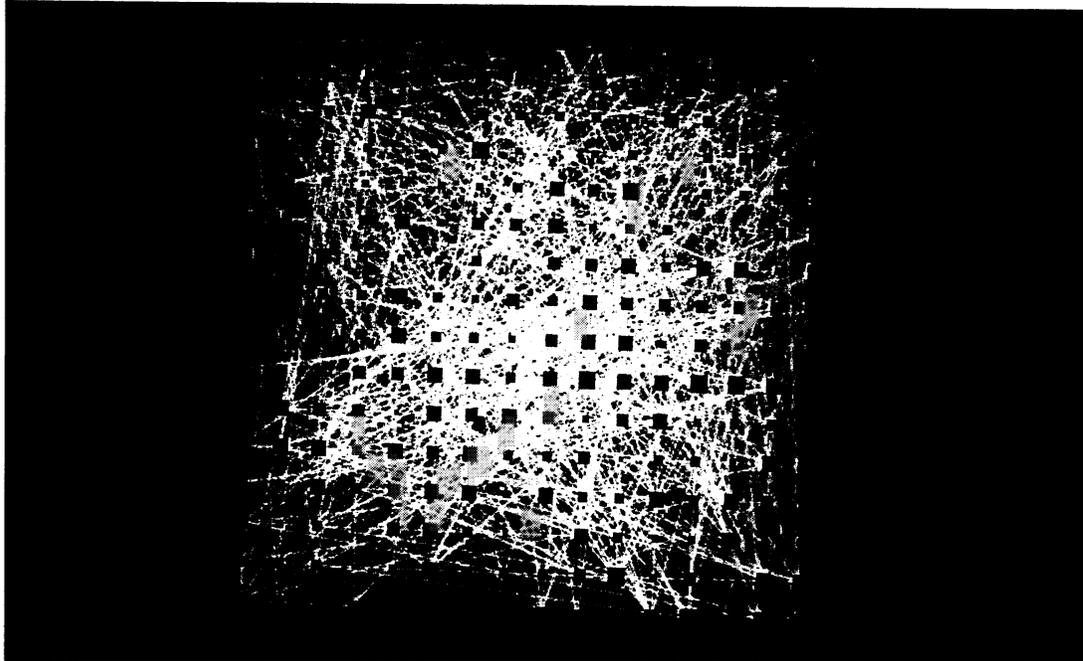


Figure 3 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. §IV -3.2.2, Figure 58). Vu qu'il y a plusieurs niveaux de potentiels d'interaction de singletons, on les a regroupés par bloc et, ensuite, on les a représentés par des carrés rouges dont la dimension est proportionnelle à la valeur du potentiel commun. Représentation des potentiels d'interaction de paires, à partir d'un certain seuil, par des barres vertes.

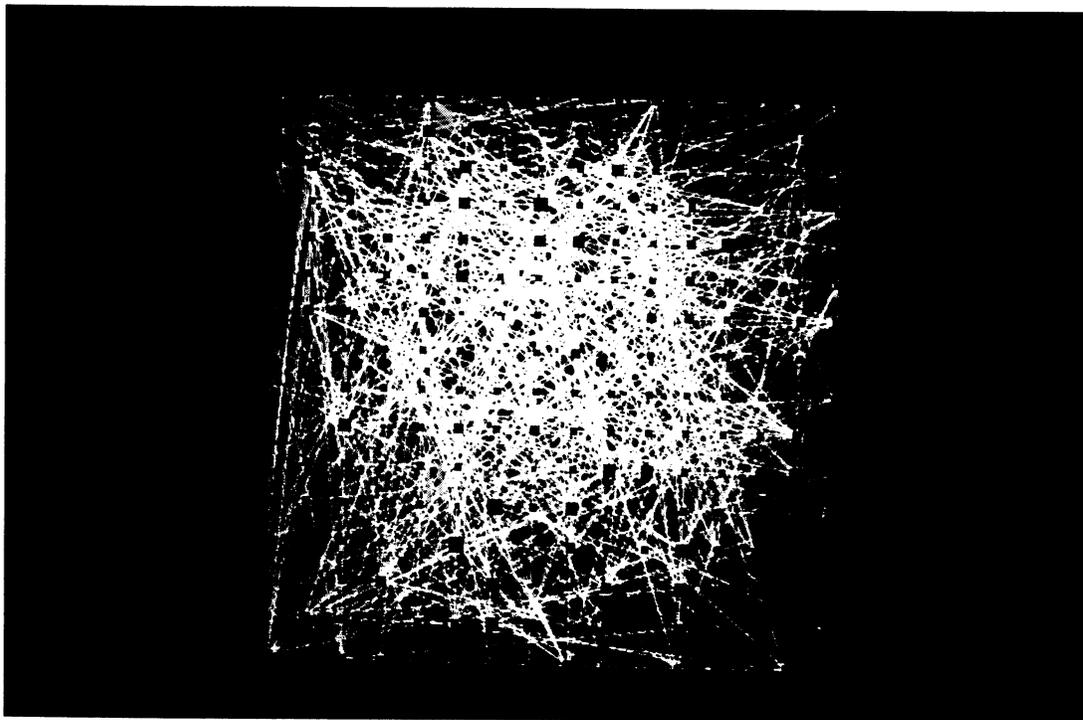


Figure 4 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération bloc-séquentiel. Evolution pendant (12000-Tcvs) itérations (cf. §IV -3.2.2, Figure 58). Vu qu'il y a plusieurs niveaux de potentiels d'interaction de singletons, on les a regroupés par bloc et, ensuite, on les a représentés par des carrés rouges dont la dimension est proportionnelle à la valeur du potentiel commun. Représentation des potentiels d'interaction de paires, à partir d'un certain seuil, par des barres vertes.

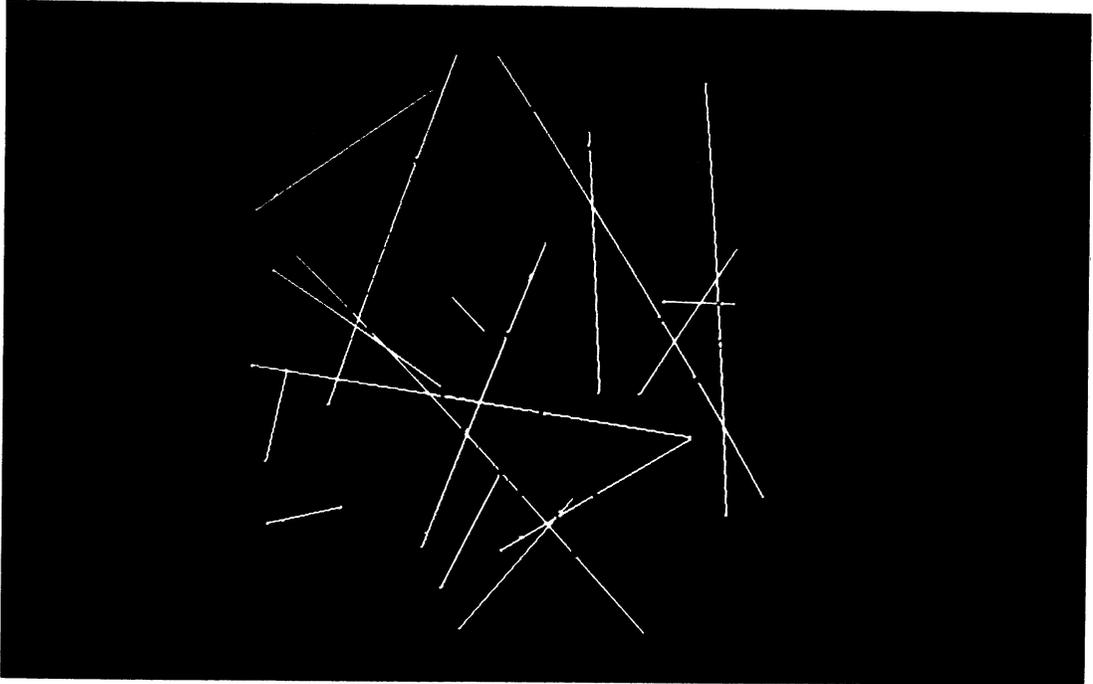


Figure 5 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. §IV -3.2.2, Figure 58). Représentation de potentiels d'interaction de singletons par des carrés rouges dont la dimension est proportionnelle à la valeur du potentiel (resp. de paires par des barres vertes).

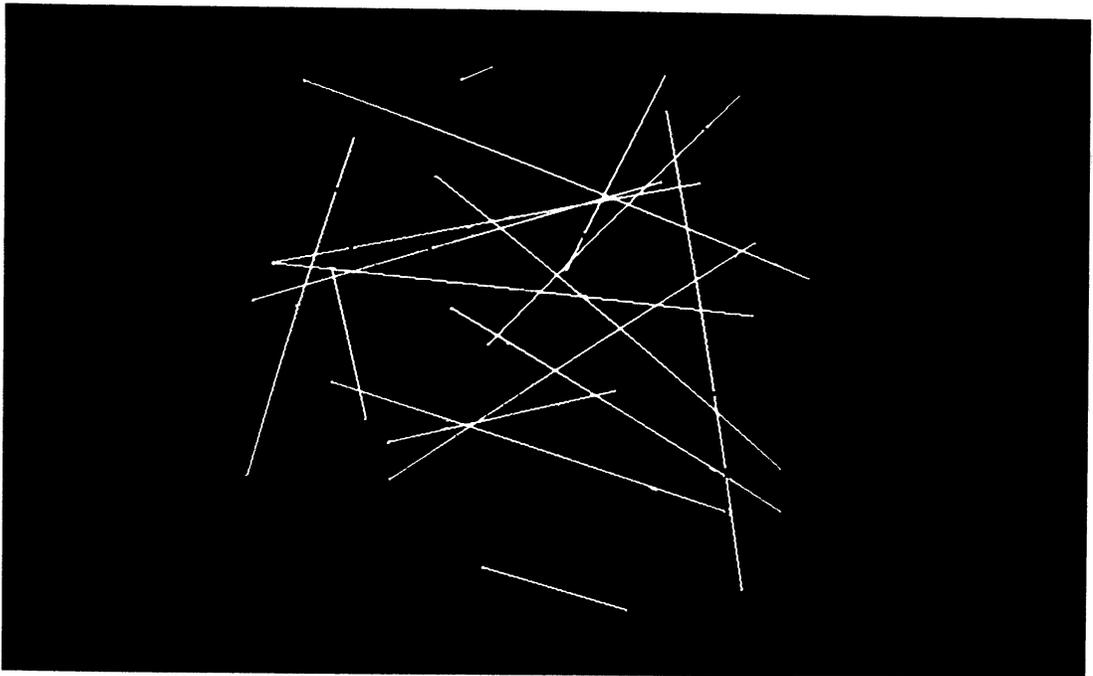


Figure 6 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Evolution pendant (12000 - Tcvs) itérations (cf. §IV -3.2.2, Figure 58). Représentation de potentiels d'interaction de singletons par des carrés rouges dont la dimension est proportionnelle à la valeur du potentiel (resp. de paires par des barres vertes).

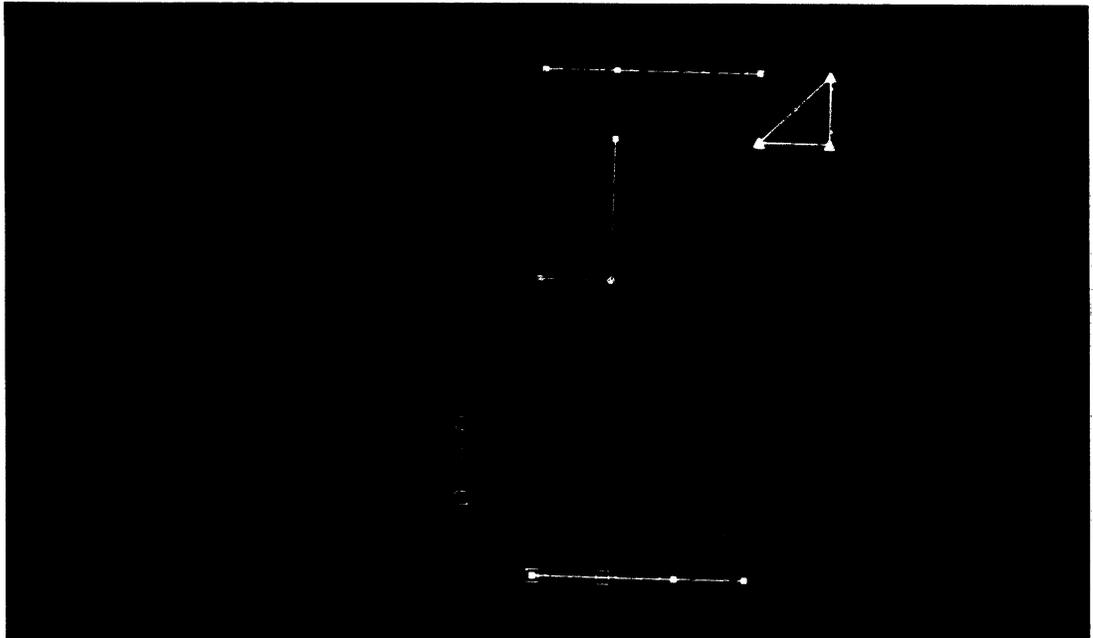


Figure 7 : Réseau de neurones de taille $N = 8 \times 8$ en mode d'itération parallèle. Evolution pendant (12000 - Tcyp) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (75) 2-cliques (en flèches rouges), (1) 3-cliques (en triangle vert), (3) 3-amas non cliques (en carrés pleins verts clairs) et (1) 4-amas non cliques (en carré vide marron).

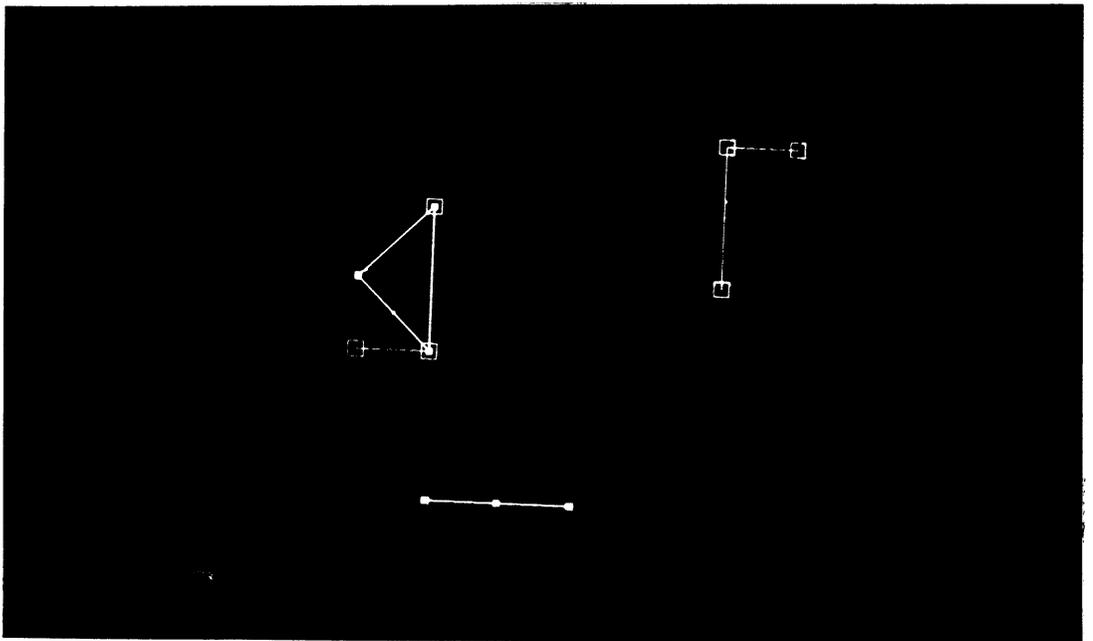


Figure 8 : Réseau de neurones de taille $N = 8 \times 8$ en mode d'itération bloc-séquentiel. Evolution pendant (12000 - Tcvs) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (87) 2-cliques (en flèches rouges), (2) 3-cliques (en carrés pleins vert) et (2) 3-amas non cliques (en carrés vides verts clairs).

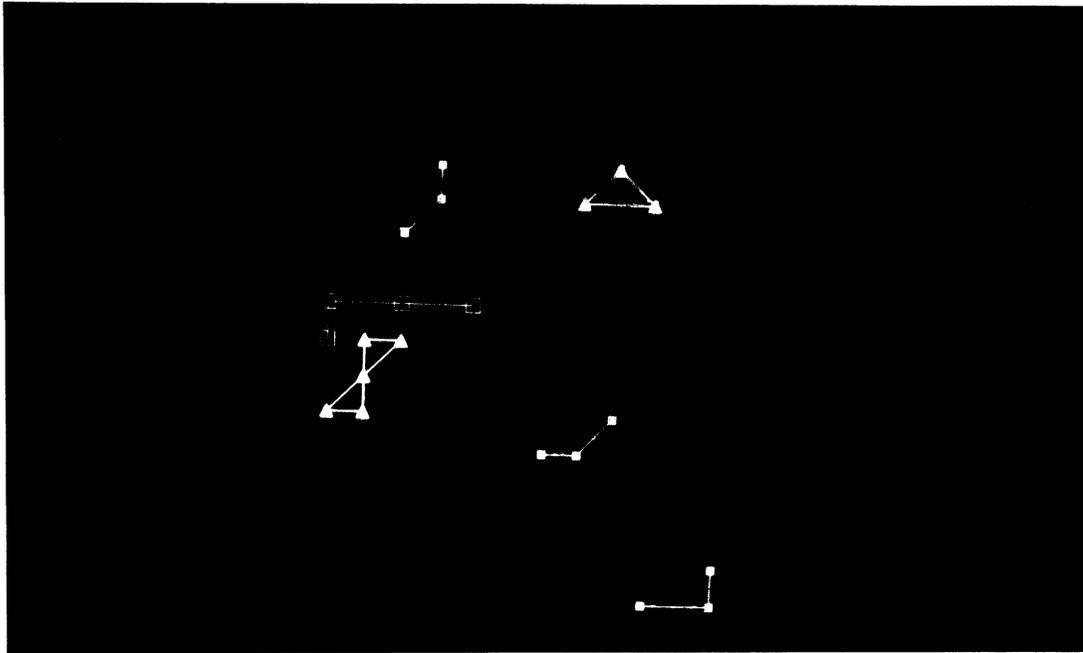


Figure 9 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération parallèle. Evolution pendant (12000 - T_{cvp}) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (309) 2-cliques (en flèches rouges), (3) 3-cliques (en triangle vert), (3) 3-amas non cliques (en carrés pleins verts clairs) et (1) 4-amas non cliques (en carré vide marron).

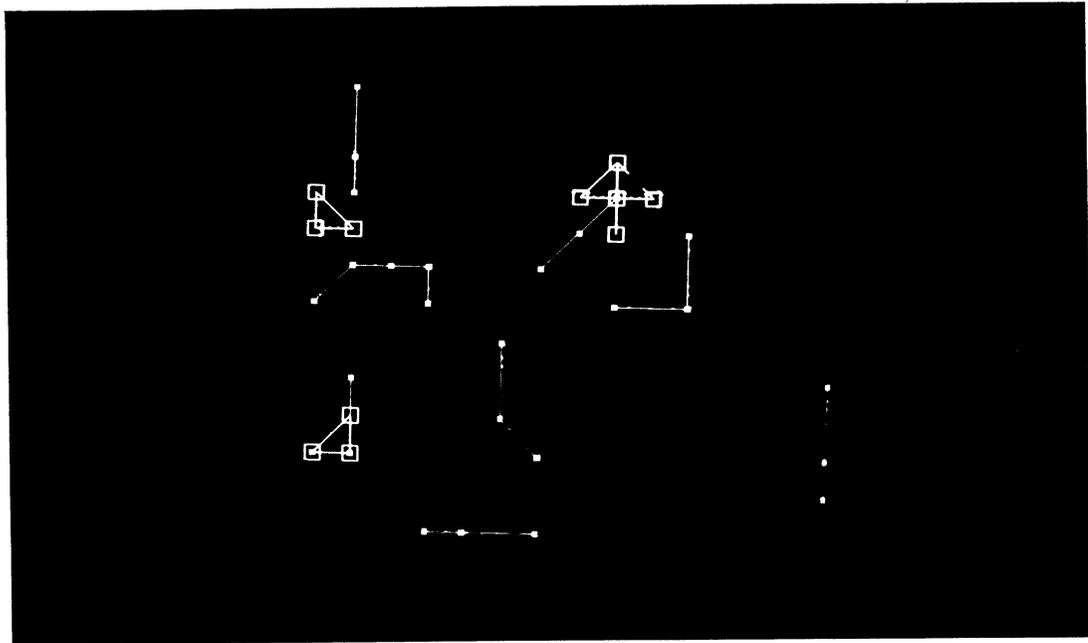


Figure 10 : Réseau de neurones de taille $N = 16 \times 16$ en mode d'itération bloc-séquentiel. Evolution pendant (12000 - T_{cvs}) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (298) 2-cliques (en flèches rouges), (4) 3-cliques (en carrés vides verts) et (9) 3-amas non cliques (en carrés pleins verts clairs).

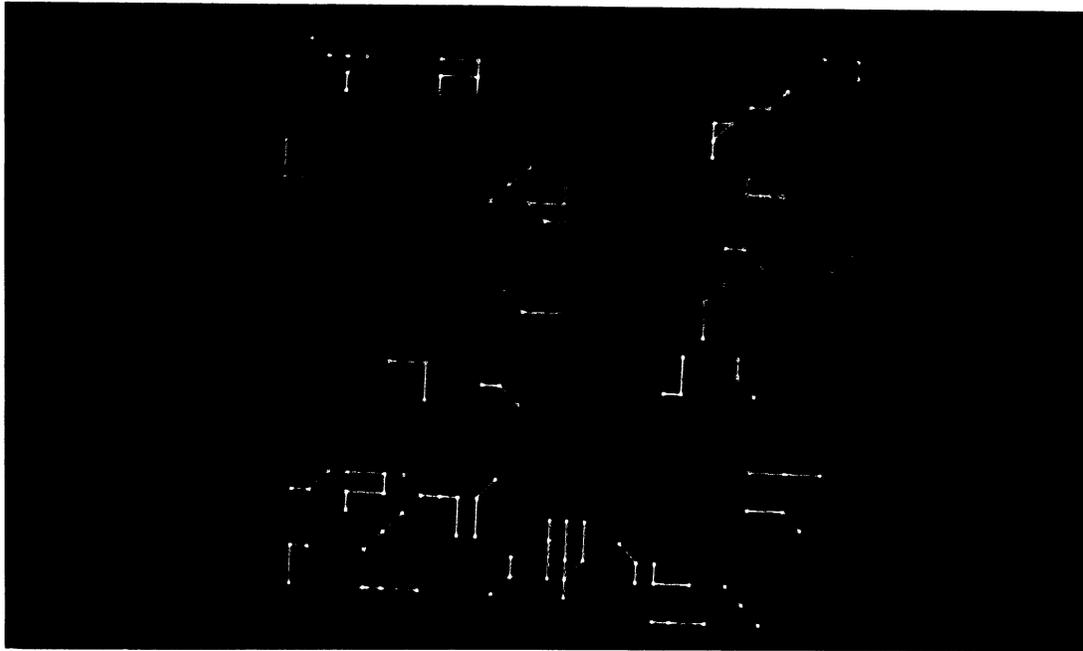


Figure 11 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération parallèle. Evolution pendant (12000 - T_{cvp}) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (9) 3-cliques (en carrés vides rouges), (40) 3-amas non cliques (en carrés pleins verts) et (2) 4-amas non cliques (en triangles vides marron).

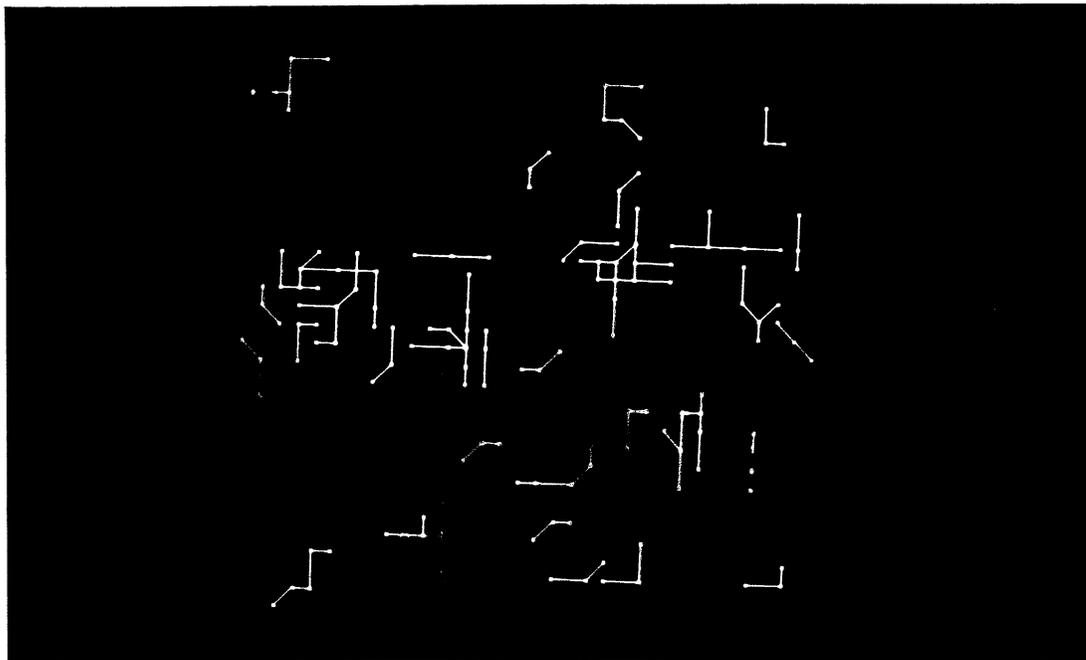


Figure 12 : Réseau de neurones de taille $N = 32 \times 32$ en mode d'itération bloc-séquentiel. Evolution pendant (12000 - T_{cvs}) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (11) 3-cliques (en carrés vides rouges), (53) 3-amas non cliques (en carrés pleins verts) et (1) 4-amas non cliques (en triangle vide marron).

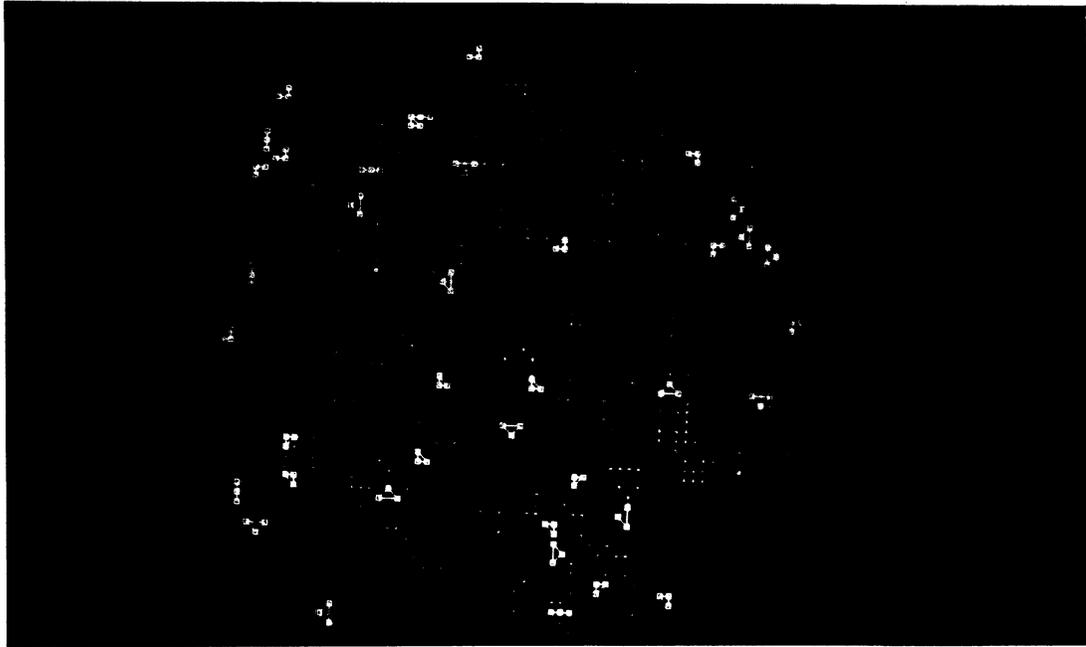


Figure 13 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération parallèle. Evolution pendant (12000-Tcvp) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (39) 3-cliques (en carrés vides verts), (143) 3-amas non cliques (en carrés pleins verts clairs), (1) 4-cliques (en triangle plein marron) et (6) 4-amas non cliques (en triangles vides rouges).

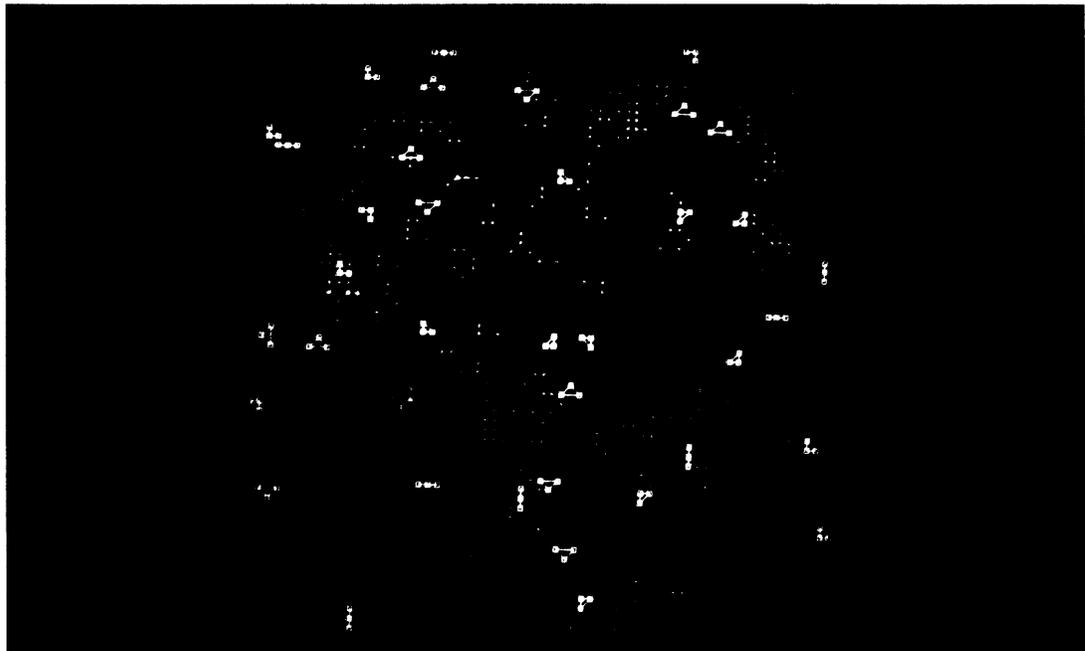


Figure 14 : Réseau de neurones de taille $N = 64 \times 64$ en mode d'itération bloc-séquentiel. Evolution pendant (12000 - Tcvp) itérations (cf. §IV -3.2.2, Figure 58). Représentation simultanée des configurations : (37) 3-cliques (en carrés vides verts), (149) 3-amas non cliques (en carrés pleins verts clairs), (1) 4-cliques (en triangle plein marron) et (8) 4-amas non cliques (en triangles vides rouges).

PARTIE (B)**Schémas des exécutions et programmes***** SCHEMAS DES EXECUTIONS DES PROGRAMMES****I - Calculs et visualisations des potentiels d'interactions :**

Cela est résumé en lançant le "BATCH" suivant :

```
$ SU / LOG=[BENAOUDA] POT.COM
```

où "POT.COM" est un ensemble de commandes suivant :

• **POT.COM :**

```
$ SET DEF D$USER 0 : [BENAOUDA]
$ COPY [BENAOUDA.HYPERCUBE.PAR32] OUT*.* [BENAOUDA]
$ SET VER ! 1ère exécution
$ ASS / USER 32PLECT.TXT SYS $ OUTPUT
$ RUN LECT
32 ! taille du réseau
12000 ! Kmax itérations
PAR ! mode d'itération
$ SET NOVER
$ DEL OUT*.* ; 1
$ PRINT 32PLECT.TXT / QUEUE = LCHU _ ANSI
$ SET VER ! 2ème exécution
$ ASS / USER 32PPOT.TXT SYS $ OUTPUT
$ RUN POTENTIELS
32 ! taille du réseau
12000 ! Kmax itérations
PAR ! mode d'itération
2669 ! Tcyp le début des tests
$ SET NOVER
$ PRINT 32PPOT.TXT / QUEUE = LCHU _ ANSI
$ RUN TRANSFERT ! 3ème exécution
$ RUN VISU_IMAGE
$ EXIT
```

Commentaires :*1° La 1ère exécution du batch :*

- *Lecture des 16 fichiers binaires " OUTIJ.BIN " provenant de l'hypercube T -40.*
- *Obtention d'un fichier binaire " SG_PAIRE.ENT " :*

Il contient successivement, jusqu'à Kmax itérations, les informations suivantes :

- *Le numéro de l'itération courante " n0_it ",*
- *Le nombre de neurones excités " nb_un ",*
- *Les coordonnées des neurones excités de la configuration d'ordre "k" où k=1,2.*

2° La 2ème exécution du batch :

- *Lecture du fichier binaire " SG_PAIRE.ENT " (généré par 1°).*
- *Obtention de 2 fichiers binaires :*

- *" JSGLTN.IMA " :*

Il contient successivement, à partir de Tcv jusqu'à la fin du fichier lecture, les coordonnées des singletons et leurs potentiels d'interaction normalisés.

- *" JPAIRE.IMA " :*

Il contient successivement, à partir de Tcv jusqu'à la fin du fichier lecture, les coordonnées des paires et leurs potentiels d'interaction normalisés.

3° La 3ème exécution du batch :

- *transformer les fichiers suivants :*

- *" JSGLTN.IMA " en " JSGLTN_FOR.IMA "*

- *" JPAIRE.IMA " en " JPAIRE_FOR.IMA "*

c'est-à-dire, que l'on a regroupé l'information des singletons (resp. des paires) en lignes et cela pour faciliter la lecture dans un programme écrit en langage "FORTRAN".

- *Obtention de 2 images (512 x 512) suivantes :*

- *" IMA_PA.DAT " : Représentation des paires par des barres,*

- *" NEURAL.DAT " : Représentation des singletons par des carrés pleins en différentes dimensions et des paires par des barres.*

II - Calculs et visualisations des cliques et amas du réseau :

Cela est résumé en lançant le "BATCH" suivant :

```
$ SU / LOG=[BENAOUDA] AMAS.COM
```

où "AMAS.COM" est un ensemble de commandes suivant :

• AMAS.COM :

```
$ SET DEF D$USER 0 : [BENAOUDA]
$ COPY [BENAOUDA.HYPERCUBE.PAR32] OUT*.* [BENAOUDA]
$ SET VER                                     ! 1ère  exécution
$ ASS / USER  32PCREA_MARKOV.TXT  SYS $ OUTPUT
$ RUN CREA_MARKOV
32                                     !  taille du réseau
12000                                !  Kmax itérations
20                                    !  l'Ordre de rééchantillonnage
PAR                                   !  mode d'itération
$ SET NOVER
$ DEL OUT*.* ; 1
$ PRINT 32PCREA_MARKOV.TXT / QUEUE = LCHU _ ANSI
$ SET VER                                     ! 2ème  exécution
$ ASS / USER  32PAMAS2.TXT  SYS $ OUTPUT
$ RUN AMAS
32                                     !  taille du réseau
12000                                !  Kmax itérations
20                                    !  l'Ordre de rééchantillonnage
2669                                  !  Tcvp le début des tests
PAR                                   !  mode d'itération
$ SET NOVER
$ PRINT 32PAMAS2.TXT / QUEUE = LCHU _ ANSI
$ SET VER                                     ! 3ème  exécution
$ ASS / USER  32PVISU_AMAS.TXT  SYS $ OUTPUT
$ RUN VISU_AMAS
32                                     !  taille du réseau
12000                                !  Kmax itérations
20                                    !  l'Ordre de rééchantillonnage
PAR                                   !  mode d'itération
$ SET NOVER
$ PRINT 32PVISU_AMAS.TXT / QUEUE = LCHU _ ANSI
$ SET VER                                     ! 4ème  exécution
$ ASS / USER  32PVISU_CLIQ.TXT  SYS $ OUTPUT
$ RUN VISU_CLIQ
32                                     !  taille du réseau
12000                                !  Kmax itérations
20                                    !  l'Ordre de rééchantillonnage
PAR                                   !  mode d'itération
$ SET NOVER
$ PRINT 32PVISU_CLIQ.TXT / QUEUE = LCHU _ ANSI
$ RUN TRANSFERT2                         ! 5ème  exécution
$ RUN TRANSFERT3
$ RUN 64 IMAGE_AMAS4
$ RUN 64 IMAGE_AMAS3
$ RUN IMA_FLECH
$ RUN IMA_FLECH2
$ EXIT
```

Commentaires :**1° La 1ère exécution du batch :**

- Lecture des 16 fichiers binaires " OUTIJ.BIN " provenant de l'hypercube T -40.
- Obtention d'un fichier binaire " MARKOV.ENT " :

Il contient successivement, jusqu'à Kmax itérations, les informations suivantes :

- *Le numéro de l'itération courante " n0_it ",*
- *Le nombre de neurones excités " nb_un ",*
- *Les coordonnées des neurones excités de la configuration d'ordre "k" où $k = 1, \dots, \text{Ordre}$.*

2° La 2ème exécution du batch :

- Lecture du fichier binaire " MARKOV.ENT " (généré par 1°).
- Obtention de 2 fichiers binaires :

- " SPTQ_CLIQ.ENT " :

Il contient successivement, à partir de Tcv jusqu'à la fin du fichier lecture, les coordonnées des configurations : n-cliques, où $n = 1, \dots, 4$, si elles existent !, i.e. les singletons, les paires, les triplets et les quadruplets en cliques, ainsi que leur nombre d'apparitions.

- " AMAS.ENT " :

Il contient successivement, à partir de Tcv jusqu'à la fin du fichier lecture, les coordonnées des configurations : n-amas, où $n \geq 3$, si elles existent !, et aussi leur nombre d'apparitions.

3° La 3ème exécution du batch :

- Lecture du fichier binaire " AMAS.ENT " (généré par 2°).
- Obtention de 2 fichiers binaires :

- " TR_AMA.IMA " :

Il contient successivement, jusqu'à la fin du fichier lecture, les coordonnées des configurations : 3-amas, ainsi que leur nombre d'apparitions.

- " QU_AMA.IMA " :

Il contient successivement, jusqu'à la fin du fichier lecture, les coordonnées des configurations : 4-amas, aussi leur nombre d'apparitions.

4° *La 4ème exécution du batch :*

- *Lecture du fichier binaire " SPTQ_CLIQ.ENT " (généré par 2°).*
- *Obtention de 2 fichiers binaires :*

- " PA.IMA " :

Il contient successivement, jusqu'à la fin du fichier lecture, les coordonnées des configurations : 2-cliques, ainsi que leur nombre d'apparitions.

- " TR.IMA " :

Il contient successivement, jusqu'à la fin du fichier lecture, les coordonnées des configurations : 3-cliques, ainsi que leur nombre d'apparitions.

- " QU.IMA " :

Il contient successivement, jusqu'à la fin du fichier lecture, les coordonnées des configurations : 4-cliques, ainsi que leur nombre d'apparitions.

5° *La 5ème exécution du batch :*

- *transformer les fichiers suivants :*

- " PA.IMA" en " PA_FOR.IMA"

- " TR.IMA" en " TR_FOR.IMA"

- " QU.IMA" en " QU_FOR.IMA"

- " TR_AMA.IMA" en " TR_AMA_FOR.IMA"

- " QU_AMA.IMA" en " QU_AMA_FOR.IMA"

même explication donnée dans (1).

- *Obtention des images (512 x 512) suivantes :*

- "IMA_PA.DAT" : *Représentation des 2-cliques par des flèches double sens,*

- "IMA_TR.DAT" : *Représentation des 3-cliques par des carrés, triangles,*

- "IMA_QU.DAT" : *Représentation des 4-cliques par des carrés, triangles,*

- "IMA_TRAMA.DAT" : *Représentation des 3-amas par des carrés, triangles,*

- "IMA_QUAMA.DAT" : *Représentation des 4-amas par des carrés, triangles,*

- "IMA_TQAMA.DAT" : *Addition des images ci-dessus sauf celle des paires,*

- "IMA_PT.DAT " : *Addition des images ci-dessus avec celle des paires.*

*** programmes :**

Ces programmes sont implémentés, en langages "C" et "FORTRAN", sur le Vax-6000 dont le système d'exploitation est VMS.

```

/*****
** LECT.C : lit les 16 fichiers binaires 'outij.bin'
** provenant de l'hypercube et reconstitue le champ de sortie
** Globale du reseau contenant que des configurations SINGLETONS
** et PAIRES, stockees dans un fichier 'SG_PAIRE.ENT' :
** - Numero d'iteration,
** - Nombre de neurones excites,
** - les coordonnees des neurones excites,
** ect...
**/*****
#include stdio

extern LIB$INIT_TIMER();
extern LIB$SHOW_TIMER();

int nb_iter[4][4]; /* Numero d'iteration courrante sur le processeurij */
int nb_unit[4][4]; /* Nombre de neurones excites sur le processeurij */
unsigned int conv[4][4][32]; /* Tableaux de conversion des ss_matrices
int k[16],l[16]; /* reperant la position vertical et horizontal
du processeur dans la grille dont la sous
matrice n'est pas vide

int sortie[128][128];
int Kmax; /* nombre total d'iterations
int taille; /* la taille du reseau global
int N; /* la taille des sous matrices dans chaque processeur*/
int nb_tot_un; /*nombre total de l a l'iteration l'tmain dans la matrice
globale du reseau
int l'tmain; /* numero d'iteration courant du reseau

/* ----- pointeurs des fichiers a lire -----
static FILE *f00,*f01,*f02,*f03,*f10,*f11,*f12,*f13,*f20,*f21,*f22,*f23,*
*f30,*f31,*f32,*f33;

/* ----- pointeur du fichier de sortie -----
static FILE *fsortie;

/*****
** fonctions et procedures
**/*****
/*
** fonction l'tmain: -calcule l'iteration courrante sur les 16
** fichiers outij.bin
**/*****
int it_min()
{
    int min;
    int i,j;
    min=nb_iter[0][0];
    for (i=0;i<=3;i++)
        for (j=0;j<=3;j++)
            if (nb_iter[i][j]<min)

```

```

        (min=nb_iter[i][j]);
    }
    return min;
}
/*****
** fonction lire_entier :
** lit un entier de type int sur le fichier outij .bin
**/*****
int lire_entier(I,J)
{
    int x;
    x=0;
    if ((I==0) && (J==0) && (feof(f00)==0)) { fread(&x,4,1,f00); }
    if ((I==0) && (J==1) && (feof(f01)==0)) { fread(&x,4,1,f01); }
    if ((I==0) && (J==2) && (feof(f02)==0)) { fread(&x,4,1,f02); }
    if ((I==0) && (J==3) && (feof(f03)==0)) { fread(&x,4,1,f03); }
    if ((I==1) && (J==0) && (feof(f10)==0)) { fread(&x,4,1,f10); }
    if ((I==1) && (J==1) && (feof(f11)==0)) { fread(&x,4,1,f11); }
    if ((I==1) && (J==2) && (feof(f12)==0)) { fread(&x,4,1,f12); }
    if ((I==1) && (J==3) && (feof(f13)==0)) { fread(&x,4,1,f13); }
    if ((I==2) && (J==0) && (feof(f20)==0)) { fread(&x,4,1,f20); }
    if ((I==2) && (J==1) && (feof(f21)==0)) { fread(&x,4,1,f21); }
    if ((I==2) && (J==2) && (feof(f22)==0)) { fread(&x,4,1,f22); }
    if ((I==2) && (J==3) && (feof(f23)==0)) { fread(&x,4,1,f23); }
    if ((I==3) && (J==0) && (feof(f30)==0)) { fread(&x,4,1,f30); }
    if ((I==3) && (J==1) && (feof(f31)==0)) { fread(&x,4,1,f31); }
    if ((I==3) && (J==2) && (feof(f32)==0)) { fread(&x,4,1,f32); }
    if ((I==3) && (J==3) && (feof(f33)==0)) { fread(&x,4,1,f33); }
    return x;
}
/*****
** procedure stock matrice : si la configuration
** globale est un singleton ou une paire, elle ecrit la position des
** neurones excites sur le fichier de sortie
**/*****
stock matrice(I,J)
{
    int nb_max;
    int p,s;
    int posv,posh;
    int compt_bit;
    int compt_ent;
    unsigned int convi[32];

    if (N > 4)
        (nb_max=(N*N/32)-1);
    else
        (nb_max=0);
    compt_ent=0;
    if ((I==0) && (J==0) && (feof(f00)==0))
        while (compt_ent<=nb_max)
            {
                fread(&conv[0][0][compt_ent],4,1,f00);
                compt_ent+=1;
            }
    if ((I==0) && (J==1) && (feof(f01)==0))

```

```

    { while (compt_ent<=nb_max)
      { fread(&conv[0][1][compt_ent], 4, 1, f01);
        compt_ent++;
      }
    }
    if ((I==0) && (J==2) && (feof(f02)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[0][2][compt_ent], 4, 1, f02);
        compt_ent++;
      }
    }
    if ((I==0) && (J==3) && (feof(f03)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[0][3][compt_ent], 4, 1, f03);
        compt_ent++;
      }
    }
    if ((I==1) && (J==0) && (feof(f10)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[1][0][compt_ent], 4, 1, f10);
        compt_ent++;
      }
    }
    if ((I==1) && (J==1) && (feof(f11)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[1][1][compt_ent], 4, 1, f11);
        compt_ent++;
      }
    }
    if ((I==1) && (J==2) && (feof(f12)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[1][2][compt_ent], 4, 1, f12);
        compt_ent++;
      }
    }
    if ((I==1) && (J==3) && (feof(f13)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[1][3][compt_ent], 4, 1, f13);
        compt_ent++;
      }
    }
    if ((I==2) && (J==0) && (feof(f20)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[2][0][compt_ent], 4, 1, f20);
        compt_ent++;
      }
    }
    if ((I==2) && (J==1) && (feof(f21)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[2][1][compt_ent], 4, 1, f21);
        compt_ent++;
      }
    }
    if ((I==2) && (J==2) && (feof(f22)==0))
    { while (compt_ent<=nb_max)
      { fread(&conv[2][2][compt_ent], 4, 1, f22);
        compt_ent++;
      }
    }
  }
  if ((I==2) && (J==3) && (feof(f23)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[2][3][compt_ent], 4, 1, f23);
      compt_ent++;
    }
  }
  if ((I==3) && (J==0) && (feof(f30)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][0][compt_ent], 4, 1, f30);
      compt_ent++;
    }
  }
  if ((I==3) && (J==1) && (feof(f31)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][1][compt_ent], 4, 1, f31);
      compt_ent++;
    }
  }
  if ((I==3) && (J==2) && (feof(f32)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][2][compt_ent], 4, 1, f32);
      compt_ent++;
    }
  }
  if ((I==3) && (J==3) && (feof(f33)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][3][compt_ent], 4, 1, f33);
      compt_ent++;
    }
  }
  compt_ent=0;
  while (compt_ent<=nb_max)
  { convl[compt_ent]= conv[I][J][compt_ent];
    compt_ent++;
  }
  /* decodage */
  p=N-1;
  s=N-1;
  if (nb_tot_un<=2)
  { /*pour des reseaux de tailles 8 x 8 */
    if (N==2)
    { for (compt_bit=1; compt_bit<=4; compt_bit++)
      { if (((convl[0])%2)==1)
        { posv=N+Itp;
          posh=N+Jts;
          fwrite(&posv, 1, fsortie);
          fwrite(&posh, 4, 1, fsortie);
          sortie[posv][posh]=1;
        }
        convl[0]/=2;
        s--;
        if (s<0)
        {

```



```

/*-----
*/
/*-----
*/
procEDURE ferme_fich :
/*-----
*/

```

```

fermefich ()
(
  fclose (f00);
  fclose (f01);
  fclose (f02);
  fclose (f03);
  fclose (f10);
  fclose (f11);
  fclose (f12);
  fclose (f13);
  fclose (f20);
  fclose (f21);
  fclose (f22);
  fclose (f23);
  fclose (f30);
  fclose (f31);
  fclose (f32);
  fclose (f33);
)

```

```

/*-----
*/
/*-----
*/
programme principal
/*-----
*/

```

```

main ()
(
  char lec[40];
  int i,j,s,p;
  int Itlast;
  int Nvide;
  int Nsgltn;
  int Npaire;
  int kk;

  Nvide=0;Nsgltn=0;Npaire=0;
  Itlast=1;
  nb_tot_un=0;
  kk=0;

```

```

printf ("
***SIMULATION D'UN RESEAU DE NEURONES***\n\n");
printf ("Lecture des fichiers de simulations outij.bin et reconstitue un\n");
printf ("fichier contenant des configurations Singletons et Paires dont le\n");
printf ("nom est : 'SG_PAIRES.ENT', genere par le programme:'lect.c'\n");

```

```

printf ("\nEnterz la taille du reseau (taille) ----> ");
scanf ("%d",&taille);
printf ("%d x %d\n",taille,taille);

printf ("Enterz le nombre d'iterations (Kmax) ----> ");
scanf ("%d",&Kmax);
printf ("%d\n",Kmax);

```

```

printf ("Enterz le type de mode d'implementation du reseau :\n");
printf ("Implementation Parallele --> Enter, 'par '\n");
printf ("Implementation Block-sequentiel --> Enter, 'besq '\n");
printf ("
scanf ("\n%s",&lect);
printf ("%s\n",lect);

```

```

N=taille/4;
ouvre_fich ();

```

```

fsortie=fopen("sg_paire.ent", "w");
for (i=0; i<=3; i++)
(
  for (j=0; j<=3; j++)
  (
    nb_iter[i][j]=lire_entier(i,j);
    nb_unit[i][j]=lire_entier(i,j);
  )
)

Itmin=it_min();
init_matrice();
/*LIB$INIT_TIMER();*/
while (Itmin<=Kmax)
(
  p=0;
  for (i=0; i<=3; i++)
  for (j=0; j<=3; j++)
  (
    if (nb_iter[i][j]==Itmin)
    (
      k[p]=i;
      l[p]=j;
      nb_tot_un+=nb_unit[i][j];
      p++;
    )
  )

  Nvide=Itmin-Itlast;
  Itlast=Itmin+1;
  fwrite (&nb_tot_un,4,1,fsortie);
  if (nb_tot_un<=2)
  (
    if (nb_tot_un==1)
    {
      Nsgltn=1;
    }
    else
    {
      Npaire=1;
    }
  )

  s=0;
  while (s<=p)
  (
    stock_matrice(k[s],l[s]);
    nb_iter[k[s]][l[s]]=lire_entier(k[s],l[s]);
    nb_unit[k[s]][l[s]]=lire_entier(k[s],l[s]);
    if (nb_iter[k[s]][l[s]]==0)
    {
      s++;
      (nb_iter[k[s]][l[s]]=30000;
    )
  )
)
/*imprime_matrice();
init_matrice();*/
nb_tot_un=0;
Itmin=it_min();
if (Itmin==30000) (break;
)
)/* fin while (itmin<Kmax) */
fermefich ();

```

```
fclose(fsortie);
printf("\n -----> Nombre total de configurations de vides = %d\n",Nvide);
printf(" -----> Nombre total de configurations de singletons = %d\n",Nsgltn);
printf(" -----> Nombre total de configurations de paires = %d\n",Npaire);
) /* fin main( ) */
```

```

/*****
/* programme potentiels.c :
/* calcule les potentiels de singleton et de paire ,les normalise
/* puis genere 2 fichiers : "Jsgltn.lma" & "Jpaire.lma"
/*****
/----- parametres du programme : -----
/* kmax :nombre d'iterations de la simulation
/* fich : fichier lu = "sg_paire.ent"
/-----
#include stdio
#define taille 5000
#define eps 1.0e-02
#include math

int Kmax; /* Nombre total d'iterations
int N; /* La taille du reseau global

int Seuil=0; /* Seuil entier du nombre d'apparitions de singletons
int Nvide; /* Nombre d'apparition de la configuration vide
int dim; /* Dimension courrante du tableau des paires

int Jsgltn[128][128]; /* Potentiels de singleton normalises
int paire[taille][4]; /* vecteur paire[s]=(i1,j1,i2,j2)
int Jpaire[taille]; /* potentiels de paire normalises

FILE *fich; /* pointeur du fichier de lecture
FILE *fsgltn,*fpaire; /* pointeurs des fichiers d'ecriture

double Js[128][128]; /* valeur reelle des pots de sgltns
double Jslast[128][128]; /* anterieure " " "
double moyenne; /* nombre moyen d'excitations
double dj;

/-----
/* fonction lire_entier :
/* lit et retourne un entier du fichier de lecture
/-----
int lire_entier()
{
    int x;
    fread(&x,4,1,fich);
    return x;
}

/-----
/* procedure pot_sgltn :
/* calcule les potentiels de singletons
/-----
pot_sgltn()
{
    int i,j;
    int nbsg2;

    nbsg2=0;
    for (i=0,i<=N-1;i++)
        for (j=0,j<=N-1;j++)
            if (Jsgltn[i][j]>Seuil)
                (Js[i][j]=log(Jsgltn[i][j]))-log(Nvide);
                nbsg2++;
}

/-----
/* procedure normalisation :
/* calcule les potentiels de singletons de paires et les normalise
/-----
normalisation()
{
    int i,j;
    int s;
    int sup;
    double Jmax,Jmin,Jpmax,Jpmin,Jp1,Jp2;
    double Jp[taille];
    int nbpaire2;

    nbpaire2=0;
    sup=28000;
    /*sup=256;*/
    Jsmax=1.;Jpmax=0.;Jpmin=100.;

    for (i=0,i<=N-1;i++)
        for (j=0,j<=N-1;j++)
            {
                if (Jsgltn[i][j]>=Jsmax)
                    { Jsmax=Jsgltn[i][j]; }
            }

    printf("Nvide= %d \t dim+1= %d \n",Nvide,dim);

    pot_sgltn();

    Jsmn = log(Jsmax)-log(Nvide);
    Jsmn = -log(Nvide);

    s=0;
    while (Jpaire[s]!=0)
        {
            Jp1=Js[paire[s][0]][paire[s][1]];
            Jp2=Js[paire[s][2]][paire[s][3]];

            if((Jp1!=0) && (Jp2!=0) )
                {
                    Jp[s]= ((-Jp1-Jp2)+log(Jpaire[s]))-log(Nvide);
                    if (Jp[s]>=Jpmax) { Jpmax=Jp[s]; }
                    if (Jp[s]<=Jpmin) { Jpmin=Jp[s]; }
                    /*printf("Jp[ %d ]= %.6f\n",s,Jp[s]);*/
                    nbpaire2++;
                }
            s++;
        }
    printf("nbpaire2 = %d\n",nbpaire2);

    printf("s+1= %d\n",s);

    printf("Jpmin= %e\t Jpmax= %e\n",Jpmin,Jpmax);

    /*if (Jpmin-Jpmax) (Jpmin=Jpmax-1; )
    dj=(Jpmax-Jpmin);
    if (dj<eps) (Jpmin=Jpmax-1; )
    printf("Jpmax-Jpmin= %e\n",Jpmax-Jpmin);*/
}

```

```

s=0;
while (Jpaire[s]!=0)
{
    if (Jp[s]!=0)
    {
        Jp[s]=(Jp[s]-Jpmin)*sup/(Jpmax-Jpmin+1);
        Jpaire[s] = Jp[s]+4000;
        /*Jpaire[s] = Jp[s];
        printf("Jpaire[ %d ] = %d\n",s,Jpaire[s]);*/
    }
    else
    {
        Jpaire[s]=0;
        s++;
    }
}
Jpaire[s] = 33333;

printf("Jmin= %e\t Jmax= %e\n",Jmin,Jmax);
/*if (Jmin==Jmax) (Jmin=Jmax-1;
dJ=(Jmax-Jmin);
if (dJ<eps) (Jmin=Jmax-1;
printf("Jmax-Jmin= %e\n",Jmax-Jmin);*/

for (i=0;i<=N-1;i++)
for (j=0;j<=N-1;j++)
{
    if (Js[i][j] != 0)
    {
        Js[i][j]=(Js[i][j]-Jmin)*sup/(Jmax-Jmin+1);
        /*Jsgltn[i][j] = Js[i][j];*/
        Jsgltn[i][j] = Js[i][j]+4000;
        /*printf("%d\t%d\t%d\n",i,j,Jsgltn[i][j]);*/
    }
    else
    {
        (Jsgltn[i][j]=0;
    }
}

/*-----*/
/*
/* stocke dans le fichier "Jsgltn.ima" les singletons et leur potentiel */
/*-----*/

stock_sgltn()
{
    int i,j,x,nbsing;
    nbsing = 0;
    fsgltn = fopen("Jsgltn.ima", "w");
    if(N==16)
    (printf("\n
for (i=0;i<=N-1;i++)
{
    printf("
for (j=0;j<=N-1;j++)
{
    if (Jsgltn[i][j]!=0)
    {
        fwrite(&i,4,1,fsgltn);
        fwrite(&j,4,1,fsgltn);
    }
}
}
)
}

s=0;
while (Jpaire[s]!=0)
{
    nbsing++;
    }
    else
    {
        printf(" ");
    }
}
if (N==16)
{
    printf("
-----\n");
}
x=77777;
printf("\n nb singletons = %d\n",nbsing);
fwrite(&x,4,1,fsgltn);
fclose(fsgltn);
}

/*-----*/
/*
/* stocke dans "Jpaire.ima" les paires et leur potentiel */
/*-----*/

stock_paire()
{
    int i,s,x,nbpaire;
    s=0; nbpaire=0;
    fpaire=fopen("Jpaire.ima", "w");
    while ( Jpaire[s]!=33333)
    {
        if (Jpaire[s]!=0)
        {
            for (i=0;i<=3;i++)
            {
                fwrite(&paire[s][i],4,1,fpaire);
                nbpaire++;
            }
            s++;
        }
        printf("nb paires = %d\n",nbpaire);
        x=33333;
        fwrite(&x,4,1,fpaire);
        fclose(fpaire);
    }
}

/*-----*/
/*
/* procedure initialisation :
/*
/* initialise les parametres globaux du programme */
/*-----*/

initialisation()
{
    int i,j;
    dim=0;Nvide=0;moyenne=0.0;
    for(i=0;i<=N-1;i++)
    for(j=0;j<=N-1;j++)
    {
        Jsgltn[i][j]=0;
    }
}

/*-----*/
/*
/* programme principal
/*-----*/

```

```

main()
{
    char lect[40];
    int i,j,s;
    int il,j1,i2,j2;
    int itlast;
    int drapeau,flag;
    int n0_it,nb_un;
    int Kl,Np;

    printf("Entrez la taille du reseau (N)
    -----> ");
    scanf("%d",&N);
    printf("%d z %d\n",N,N);

    printf("Entrez le nombre total d'iterations (Kmax)
    -----> ");
    scanf("%d",&Kmax);
    printf("%d\n",Kmax);

    printf("Entrez le type de mode d'implementation du reseau :\n");
    printf("Implementation Parallele --> Entrez 'par '\n");
    printf("Implementation Block-sequentiel --> Entrez 'bseq '\n\n");
    scanf("%s",&lect);
    printf("%s\n",lect);

    printf("Entrez le debut d'estimation (Kl)
    -----> ");
    scanf("%d",&Kl);
    printf("%d\n",Kl);

    Itlast=1;flag=0;
    Np=0;
    initialisation();
    for (i=0;i<N-1;i++)
        for (j=0;j<N-1;j++)
            {Jslast[i][j]=0.0;}

    fich=fopen("sg_paire.ent","r");
    n0_it=lire_entier();
    Nvide=n0_it-Itlast;
    while ( (n0_it<Kmax) && (feof(fich)==0) )
        {
            nb_un=lire_entier();
            moyenne+nb_un;
            if (nb_un==1)
                {
                    il=lire_entier();
                    j1=lire_entier();
                    Jsgltn[il][j1]+=1;
                }
            if (nb_un==2)
                {
                    il=lire_entier();
                    j1=lire_entier();
                    i2=lire_entier();
                    j2=lire_entier();
                    s=0;
                    drapeau=0;
                    while (s<dim)
                        {
                            if ( ((paire[s][0]==i1)&&(paire[s][1]==j1)&&(paire[s][2]==i2)
                            &&(paire[s][3]==j2)) || ((paire[s][0]==i2)&&(paire[s][1]==j2)&&(paire[s][2]==i1)
                            &&(paire[s][3]==j1)) )
                                {
                                    Np++;
                                    Jpaire[s]++;
                                    drapeau=1;
                                    break;
                                }
                            else
                                {
                                    s++;
                                }
                            if (drapeau==0)
                                {
                                    paire[dim][0]=il;
                                    paire[dim][1]=j1;
                                    paire[dim][2]=i2;
                                    paire[dim][3]=j2;
                                    Jpaire[dim]=1;
                                    dim++;
                                }
                            if (drapeau==0)
                                {
                                    Itlast=n0_it;
                                    n0_it=lire_entier();
                                    Nvide=n0_it-Itlast-1;
                                    if (flag==0)
                                        {
                                            if (n0_it>=Kl) (initialisation();flag=1;Np=0;)}
                                        }
                                    /* fin while (n0_it < Kmax) */
                                }
                            if (drapeau==1) (dim+=1;
                                for (i=0;i<3;i++) { paire[dim][i]=0; }
                                Jpaire[dim]=0;
                                printf("Np = %d\n",Np);
                                printf("normalisation\n"); normalisation();
                                printf("stock sgltn\n"); stock_sgltn();
                                printf("stock paire\n"); stock_paire();
                                moyenne/=n0_it-Kl;
                                printf("moyenne = %e\n",moyenne);
                                fclose(fich);
                            }
                        }
                    }
}

```

```
c      transcr.for
      subroutine escrifor1(im,tab)
      integer i
      integer tab(10000),dim5
      character *40 fich
      fich='Jpaire.for.ima'
      open(1,file=fich,access='sequential',status='unknown',err=101,
      form='unformatted')
      dim5=im/5
      type *,im,dim5',im,dim5
      do i=1,dim5
      write(1)(tab((i-1)*5+k),k=1,5)
      end do
      goto 110
      type *,'erreur ouverture 1'
101      close(1)
110      return
      end
      subroutine escrifor2(im,tab)
      integer i,dim3
      integer tab(10000)
      character *40 fich
      fich='Jsqitn.for.ima'
      open(2,file=fich,access='sequential',status='unknown',err=102,
      form='unformatted')
      dim3=im/3
      type *,im,dim3',im,dim3
      do i=1,dim3
      write(2)(tab((i-1)*3+k),k=1,3)
      end do
      goto 120
      type *,'erreur ouverture 2'
102      close(2)
120      return
      end
```

```

/*****
** TRANSFERT.C :
** transforme l'écriture d'un fichier en un fichier equivalent
** mais regroupant les informations ligne par ligne.
**
** Ce programme est linke avec le fichier fortran "transcr.for".
** La compilation se fait donc comme suit :
** * $ fort transcr
** * $ link transfert,transcr.obj
**
**/*****
#include stdio
extern ecrifor1();
extern ecrifor2();

main()
{
FILE *fich;
int xi;
int t[25000];
int s;

s=0;x=0;

fich=fopen("Jpaire.ima","r");
while ( xi=33333 )
{
fread(&x,4,1,fich);
t[s]=x;
s++;
}
s--;
fclose(fich);
printf("Nombre de paire(normalises)= %d\n",s);
ecrifor1(&s,t);

s=0;x=0;

fich=fopen("Jsgltn.ima","r");
while (xi=77777)
{
fread(&x,4,1,fich);
t[s]=x;
s++;
}
s--;
fclose(fich);
printf("Nombre de singletons (normalises)= %d\n",s);
ecrifor2(&s,t);
}

```



```

c* tri par ordre croissant
  minl=tab(1)
  ii=1
  do i=1, compt-1
    do j=i, compt
      if (tab(j) .le. minl) then
        minl=tab(j)
        ii=j
      endif
    enddo
  enddo
  tempon=tab(ii)
  tab(ii)=tab(i)
  tab(i)=tempon
  minl=tab(i+1)
  enddo

c*
c* tableau trié
  type *, ' '
  type *, 'tri par ordre croissant'
  do i=1, compt
    type *, 'tab(' , i, ')=' , tab(i)
  enddo

c* réouverture du fichier lecture 'fsging_in'
  open (3, file=fsging_in, err=102, status='old', form='unformatted')

2  read (3, end=2000) is1, js1, ival
   rj1=js1+1
   rj1=js1+1
   ival = abs(ival)
   c  if(ival.lt.30000) then
     do i=1, compt
       if(ival .eq. tab(i)) then
         if(ival .lt. (tab(i)+6000)) then
           irayon = 2*i
           ival = 80
           call cgl$move abs 2(rj1, rj1)
           call cgl$quadrilatre ssint(rj1, rj1, irayon)
           call cgl$quadrivide_ssint(rj1, rj1, irayon)
           go to 2
         endif
       enddo
     endif
   enddo
   go to 2
2000 continue

c*-----*
c* normalisation de l'image des singletons
c*-----*

20  minl= image(1,1)
   maxl= image(1,1)
   do j=1, 512
     do i=1, 512
       if (image(i,j) .gt. 0) then
         minl=min(minl, image(i,j))
         maxl=max(maxl, image(i,j))
       endif
     enddo
   enddo
   minl=maxl
   type *, 'min,max', minl, maxl
   r=63./ (maxl-minl+1)

```

```

do i=1, 512
  do j=1, 512
    if (image(i,j) .gt. 0) then
      image(i,j) = (image(i,j) - minl) * r + 124 ! 54
    endif
  enddo
enddo
close(3)

c*-----*
c* addition de l'image "singletons" et l'image "paires"
c*-----*
file_in = 'ima pa'
open (1, file=file_in, access='sequential', status='old', err=104,
     form='unformatted', recl=128, recortype='fixed',
     type *, 'ok, fichier', file_in, 'ouvert')

do i=1, 512
  read (1, end=5000) res
  do j=1, 512
    m=res(j)
    if (image(i,j) .eq. 0) then
      image(i,j) = m
    else
      image(i,j) = image(i,j) ! +64
    endif
  enddo
enddo
go to 5
5000 continue
c*-----*
c  ecriture du fichier : neural.dat

do i=1, 512
  do j=1, 512
    m=image(i,j)
    res(j) = m
  enddo
  write(4) res
enddo
close(4)

3  continue
  call exit

c*-----*
c  messages d 'erreur
c*-----*
104 type *, 'pb ouverture fichier sortie neural.dat '
103 type *, 'pb ouverture fichier sortie ima_pa.dat '
102 type *, 'pb ouverture fichier singletons'
101 type *, 'pb ouverture fichier paires'
end

```

```

/*****
/ CREA_MARKOV.C : lit les 16 fichiers binaires 'outij.bin'
/ provenant de l'hypercube et reconstitue le champ de sortie
/ globale du reseau contenant que des configurations SINGLETONS,
/ PAIRES, TRIPLES, QUADRUPLTS, ect... jusqu'a Ordre-UPLETS :
/ stockees dans un fichier ' MARKOV.ENT ', tels que :
/ - Numero d'iteration,
/ - Nombre de neurones excites,
/ - les coordonnees des neurones excites,
/ ect...
/ *****/
#include stdio

int nb_iter[4][4]; /* Numero d'iteration courrante sur le processeurij */
int nb_unit[4][4]; /* Nombre de neurones excites sur le processeurij */
unsigned int conv[4][4][32]; /* Tableaux de conversion des ss_matrices */
int k[16],l[16]; /* reperant la position vertical et horizontal
                  du processeur dans la grille dont la sous
                  matrice n'est pas vide */
int Kmax; /* nombre total d'iterations */
int N ; /* la taille des sous matrices dans chaque processeur*/
int taille; /* la taille du reseau global */
int sortie[128][128];
int Ordre;
int nb_tot_un; /*nombre total de l a l'iteration l'tmin dans la matrice
               globale du reseau */
/***** pointeurs des fichiers a lire *****/
static FILE *f00,*f01,*f02,*f03,*f10,*f11,*f12,*f13,*f20,*f21,*f22,*f23,*f30,*f31,*f32,*f33;
/***** pointeur du fichier de sortie *****/
static FILE *fsortie;
int l'tmin; /* l'iteration courant du reseau */
/***** fonctions et procedures *****/
/*****
/ fonction l'tmin: -calcule l'iteration courrante sur les 16
/ fichiers outij.bin
/ *****/
int l't_min()
{
    int min;
    int i,j;
    min=nb_iter[0][0];
    for (i=0;i<=3;i++)
        for (j=0;j<=3;j++)
            if (nb_iter[i][j]<min)
                (min=nb_iter[i][j]);
}

```

```

        return min;
    }
    /***** fonction lire entier *****/
    /* lit un entier de type int sur le fichier outij .bin */
    /*****/
    int lire_entier(I,J)
    {
        int z;
        x=0;
        if (I==0) && (J==0) && (feof(f00)==0) ( fread(&x,4,1,f00);)
        if (I==0) && (J==1) && (feof(f01)==0) ( fread(&x,4,1,f01);)
        if (I==0) && (J==2) && (feof(f02)==0) ( fread(&x,4,1,f02);)
        if (I==0) && (J==3) && (feof(f03)==0) ( fread(&x,4,1,f03);)
        if (I==1) && (J==0) && (feof(f10)==0) ( fread(&x,4,1,f10);)
        if (I==1) && (J==1) && (feof(f11)==0) ( fread(&x,4,1,f11);)
        if (I==1) && (J==2) && (feof(f12)==0) ( fread(&x,4,1,f12);)
        if (I==1) && (J==3) && (feof(f13)==0) ( fread(&x,4,1,f13);)
        if (I==2) && (J==0) && (feof(f20)==0) ( fread(&x,4,1,f20);)
        if (I==2) && (J==1) && (feof(f21)==0) ( fread(&x,4,1,f21);)
        if (I==2) && (J==2) && (feof(f22)==0) ( fread(&x,4,1,f22);)
        if (I==2) && (J==3) && (feof(f23)==0) ( fread(&x,4,1,f23);)
        if (I==3) && (J==0) && (feof(f30)==0) ( fread(&x,4,1,f30);)
        if (I==3) && (J==1) && (feof(f31)==0) ( fread(&x,4,1,f31);)
        if (I==3) && (J==2) && (feof(f32)==0) ( fread(&x,4,1,f32);)
        if (I==3) && (J==3) && (feof(f33)==0) ( fread(&x,4,1,f33);)
        return z;
    }
    /*****/
    /* procedure stock matrice :
    /* ecrit la position de neurones excites sur le fichier de
    /* sortie
    /*****/
    stock_matrice(I,J)
    {
        int nb_max; /* la sous-matrice est codee dans un fichier
                   binaire sur N*N bits , on recupere ce
                   codage dans N*N/32 entiers */
        int p,s;
        int posv_posh;
        int compt_bit;
        int compt_ent;
        unsigned int convl[32];
        if (N > 4)
            (nb_max=(N*N/32)-1;)
        else
            (nb_max=0;)
        compt_ent=0;
        if ((I==0) && (J==0) && (feof(f00)==0))
            ( while (compt_ent<=nb_max)
              {
                fread(&convl[0][0][compt_ent],4,1,f00);
                compt_ent++;
              }
            )
    }

```

```

    compt_ent+=1;
  }
}

if ((I==2) && (J==3) && (feof(f23)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[2][3][compt_ent],4,1,f23);
      compt_ent+=1;
    }
  }

if ((I==3) && (J==0) && (feof(f30)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][0][compt_ent],4,1,f30);
      compt_ent+=1;
    }
  }

if ((I==3) && (J==1) && (feof(f31)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][1][compt_ent],4,1,f31);
      compt_ent+=1;
    }
  }

if ((I==3) && (J==2) && (feof(f32)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][2][compt_ent],4,1,f32);
      compt_ent+=1;
    }
  }

if ((I==3) && (J==3) && (feof(f33)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[3][3][compt_ent],4,1,f33);
      compt_ent+=1;
    }
  }

compt_ent=0;
while (compt_ent<=nb_max)
  { conv[compt_ent]= conv[I][J][compt_ent];
    compt_ent+=1;
  }

/* decodage */

p=N-1;
s=N-1;
if (nb_tot_un<=ordre)
  { /*pour des reseaux de tailles 8 x 8 */
    if (N==2)
      { for (compt_bit=1;compt_bit<=4;compt_bit++)
        { if (((conv[0])%2)==1)
          { posv=N*I+p;
            posh=N*J+s;
            fwrite(&posv,4,1,fsortie);
            fwrite(&posh,4,1,fsortie);
          }
          conv[0]/=2;
          s--;
          if (s<0)

```

```

  }
  { while (compt_ent<=nb_max)
    { fread(&conv[0][1][compt_ent],4,1,f01);
      compt_ent+=1;
    }
  }

if ((I==0) && (J==2) && (feof(f02)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[0][2][compt_ent],4,1,f02);
      compt_ent+=1;
    }
  }

if ((I==0) && (J==3) && (feof(f03)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[0][3][compt_ent],4,1,f03);
      compt_ent+=1;
    }
  }

if ((I==1) && (J==0) && (feof(f10)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[1][0][compt_ent],4,1,f10);
      compt_ent+=1;
    }
  }

if ((I==1) && (J==1) && (feof(f11)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[1][1][compt_ent],4,1,f11);
      compt_ent+=1;
    }
  }

if ((I==1) && (J==2) && (feof(f12)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[1][2][compt_ent],4,1,f12);
      compt_ent+=1;
    }
  }

if ((I==1) && (J==3) && (feof(f13)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[1][3][compt_ent],4,1,f13);
      compt_ent+=1;
    }
  }

if ((I==2) && (J==0) && (feof(f20)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[2][0][compt_ent],4,1,f20);
      compt_ent+=1;
    }
  }

if ((I==2) && (J==1) && (feof(f21)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[2][1][compt_ent],4,1,f21);
      compt_ent+=1;
    }
  }

if ((I==2) && (J==2) && (feof(f22)==0))
  { while (compt_ent<=nb_max)
    { fread(&conv[2][2][compt_ent],4,1,f22);

```



```
nb_unit[k[s]][l[s]]=lire_entier(k[s],l[s]);
if (nb_iter[k[s]][l[s]]==0)
  { nb_iter[k[s]][l[s]]=30000;}
  s++;
}
nb_tot_un=0;
s=0;
t++;
itmin=it_min();
/*printf("\n itmin = %d\n",itmin);*/
if (itmin==30000) ( break;
)
fermeFich();
fclose(fsortie);
T[0]=Nvide;
printf("\n\n");
printf(" Nbre iter(config non nulles) = %d\n",t);
printf(" La somme des un      = %.2f\n",som);
printf(" N Nombre moyen de l  = %.6f\n\n",som/(t+Nvide));
for(i=0;i<=Ordre;i++)
  (printf("\n T[ %d ] = %d",i,T[i]);
)
```



```

il=0;
for (i=0;i<=nb-1;i++)
  (
    ii=0;
    flag=0;
    test=0;
    for (j=0;j<=nb-1;j++)
      (
        if ( (T[Tres[i]][Tres[j]]<=2) && (j>1) )
          (
            test+=1;
            pos[ii]=Tres[j];
            ii++;
          )
      )
    if (test==1)
      (
        tl=0;
        while (tl<=nb-1)
          (
            if ( (T[pos[0]][Tres[tl]]>2) && (T[Tres[i]][Tres[tl]]>2)
              && (Tres[tl]==pos[0]) && (Tres[tl]==Tres[i]) )
              (
                tl++;
              )
            if (flag==nb-2)
              (
                cliq[2]+=1;
                Amas[2]+=1;
                Tiso[ii]=Tres[i];
                Tiso[ii+1]=pos[0];
                ii+=2;
                n0 it=Ordre*(n0 lu-1)+s;
                fwrite(&n0_it,4,1,fichout);
                n_un+=2;
                fwrite(&n_un,4,1,fichout);
                fwrite(&n_un,4,1,fichout);
                write(&neurone[Tres[i]][0],4,1,fichout);
                write(&neurone[Tres[i]][1],4,1,fichout);
                write(&neurone[Tres[i]][2],4,1,fichout);
                write(&neurone[pos[0]][0],4,1,fichout);
                write(&neurone[pos[0]][1],4,1,fichout);
                s++;
              )
            )
          )
      )
    ) /* for (i=0;i<=nb-1;i++) */
for (i=0;i<=nb-1;i++)
  (
    Tamp[i]=Tres[i];
  )
i2=0;
for (i=0;i<=nb-1;i++)
  (
    bib=0;
    j=0;
    while (j<=il-1)
      (
        if (Tamp[i]==Tiso[j])
          (
            bib+=1;break;
          )
        j++;
      )
    if (bib==0)
      (
        Tres[i2]=Tamp[i];i2++;
      )
  )
) /* fin amas_2() */
-----
/*
 *      procedure amas_3() :
 *      cette procedure stocke les 3-cliques
 *
 */
amas_3()
(
  int i,j,s,ii,tl;
  int test;
  int flag;
  static int bib;
  int n_un;

  ii=0;
  for (i=0;i<=nb-1;i++)
    (
      ii=0;
      flag=0;
      test=0;
      for (j=0;j<=nb-1;j++)
        (
          if ( (T[Tres[i]][Tres[j]]<=2) && (j>1) )
            (
              test+=1;
              pos[ii]=Tres[j];
              ii++;
            )
          if (test==2)
            (
              if ( (T[pos[0]][pos[i]]<=2) )
                (
                  tl=0;
                  while (tl<=nb-1)
                    (
                      if ( (T[pos[0]][Tres[tl]]>2) && (T[pos[1]][Tres[tl]]>2)
                        && (T[Tres[i]][Tres[tl]]>2)
                        && (T[Tres[i]][Tres[tl]]>2)
                        && (flag==1) )
                        (
                          tl++;
                        )
                      if (flag==nb-3)
                        (
                          cliq[3]+=1;
                          Amas[3]+=1;
                          Tiso[ii]=Tres[i];
                          Tiso[ii+1]=pos[0];
                          Tiso[ii+2]=pos[1];
                          ii+=3;
                          n0 it=Ordre*(n0 lu-1)+s;
                          fwrite(&n0_it,4,1,fichout);
                          n_un+=3;
                          fwrite(&n_un,4,1,fichout);
                          write(&neurone[Tres[i]][0],4,1,fichout);
                          write(&neurone[Tres[i]][1],4,1,fichout);
                          write(&neurone[pos[0]][0],4,1,fichout);
                          write(&neurone[pos[0]][1],4,1,fichout);
                          write(&neurone[pos[1]][0],4,1,fichout);
                          write(&neurone[pos[1]][1],4,1,fichout);
                          s++;
                        )
                      )
                    )
                )
            )
          )
        )
      )
    )
)

```

```

) /* for (i=0;i<nb-1;i++) */
for(i=0;i<nb-1;i++)
{
    Tmp[i]=Tres[i];
}
i2=0;
for(i=0;i<nb-1;i++)
{
    bib=0;
    j=0;
    while(j<=i1-1)
    {
        if (Tmp[i]==Tiso[j])
        {
            bib=1;
            break;
        }
        j++;
    }
    if (bib==0)
    {
        Tres[i2]=Tmp[i];
        i2++;
    }
}
} /* fin amas_3() */

/*-----*/
/*
procedure amas_4 (
cette procedure stocke les 4-cliques
*/
/*-----*/
amas_4 (
int i,j,s,i1,t1;
int test;
int flag;
int n_un;
static int bib;

i1=0;
for(i=0;i<nb-1;i++)
{
    i1=0;
    flag=0;
    test=0;
    for (j=0;j<nb-1;j++)
    {
        if ( ( T[Tres[i]][Tres[j]]<=2) && (j>i) )
        {
            test+=1;
            pos[i1]=Tres[j];
            i1++;
        }
    }
    /*printf("test = %d\ttest1 = %d\n",test,test1);*/

    if (test==3)
    {
        if( ( T[pos[0]][pos[1]]<=2) && ( T[pos[0]][pos[2]]<=2)
        && ( T[pos[1]][pos[2]]<=2) )
        {
            t1=0;
            while (t1<nb-1)
            {
                if( ( T[pos[0]][Tres[t1]]>2) && ( T[pos[1]][Tres[t1]]>2)
                && ( T[pos[2]][Tres[t1]]>2) && ( T[Tres[i]][Tres[t1]]>2) )
                {
                    flag+=1;
                    t1++;
                }
            }
        }
    }
}
}

) /* for (i=0;i<nb-1;i++) */
{
    cliq[4]=1;
    Amas[4]=1;
    Tiso[i1]=Tres[i];
    Tiso[i1+1]=pos[0];
    Tiso[i1+2]=pos[1];
    Tiso[i1+3]=pos[2];
    i1+=4;
    n0.it=Ordre*(n0.lu-1)+s;
    fwrite(&n0_it,4,1,fichout);
    n_un=4;
    fwrite(&n_un,4,1,fichout);
    fwrite(&neurone[Tres[i]][0],4,1,fichout);
    fwrite(&neurone[Tres[i]][1],4,1,fichout);
    fwrite(&neurone[pos[0]][0],4,1,fichout);
    fwrite(&neurone[pos[0]][1],4,1,fichout);
    fwrite(&neurone[pos[1]][0],4,1,fichout);
    fwrite(&neurone[pos[1]][1],4,1,fichout);
    fwrite(&neurone[pos[2]][0],4,1,fichout);
    fwrite(&neurone[pos[2]][1],4,1,fichout);
    s++;
}
}
) /* for (i=0;i<nb-1;i++) */
{
    Tmp[i]=Tres[i];
}
for(i=0;i<nb-1;i++)
{
    i2=0;
    for(i=0;i<nb-1;i++)
    {
        bib=0;
        j=0;
        while(j<=i1-1)
        {
            if (Tmp[i]==Tiso[j])
            {
                bib=1;
                break;
            }
            j++;
        }
        if (bib==0)
        {
            Tres[i2]=Tmp[i];
            i2++;
        }
    }
}
} /* fin amas_4() */

/*-----*/
/*
procedure les_amas (
stocke les neurones excites en cliques et en amas.
*/
/*-----*/
les_amas (
int i,j;
s=1;
for (i=0;i<nb-1;i++)
for (j=0;j<nb-1;j++)
{
    T[i][j]=d_mahattan(neurone[i][0],neurone[i][1],neurone[j][0],neurone[j][1]);
}
s=1;
}
}

```

```

) /* for (i=0;i<nb-1;i++) */
for(i=0;i<nb-1;i++)
{
    Tmp[i]=Tres[i];
}
i2=0;
for(i=0;i<nb-1;i++)
{
    bib=0;
    j=0;
    while(j<=i1-1)
    {
        if (Tmp[i]==Tiso[j])
        {
            bib=1;
            break;
        }
        j++;
    }
    if (bib==0)
    {
        Tres[i2]=Tmp[i];
        i2++;
    }
}
} /* fin amas_3() */

/*-----*/
/*
procedure amas_4 (
cette procedure stocke les 4-cliques
*/
/*-----*/
amas_4 (
int i,j,s,i1,t1;
int test;
int flag;
int n_un;
static int bib;

i1=0;
for(i=0;i<nb-1;i++)
{
    i1=0;
    flag=0;
    test=0;
    for (j=0;j<nb-1;j++)
    {
        if ( ( T[Tres[i]][Tres[j]]<=2) && (j>i) )
        {
            test+=1;
            pos[i1]=Tres[j];
            i1++;
        }
    }
    /*printf("test = %d\ttest1 = %d\n",test,test1);*/

    if (test==3)
    {
        if( ( T[pos[0]][pos[1]]<=2) && ( T[pos[0]][pos[2]]<=2)
        && ( T[pos[1]][pos[2]]<=2) )
        {
            t1=0;
            while (t1<nb-1)
            {
                if( ( T[pos[0]][Tres[t1]]>2) && ( T[pos[1]][Tres[t1]]>2)
                && ( T[pos[2]][Tres[t1]]>2) && ( T[Tres[i]][Tres[t1]]>2) )
                {
                    flag+=1;
                    t1++;
                }
            }
        }
    }
}
}
}

```



```

for (i=0;i<=50;i++)
  ( Amas[i]=0; )
for (i=0;i<=5;i++)
  ( cliq[i]=0; )
for (i=0;i<=150;i++)
  ( Tiso[i]=200;Tres[i]=200;Tamp[i]=200; )
flat=0;
no_lu=lire_entier();
while( (no_lu<=Kmax) && (feof(fichin)!=0) )
  ( nb=lire_entier();
    if (nb<=Ordre)
      for (i=0;i<=nb-1;i++)
        (
          neurone[i][0]=lire_entier();
          neurone[i][1]=lire_entier();
          /*sortie[neurone[i][0]][neurone[i][1]]=1;*/
        )
        /*imprime_mat();
        init_mat();*/
        if (flat==1) (les_amas());
        if (flat==0)
          (
            if (no_lu>=K1)
              (
                flat=1;
              )
            )
          )
        else
          (
            if (flat==1)
              (
                no_it=Ordre*(no_lu-1)+1;
                fwrite(&no_it,4,1,fichout);
                fwrite(&nb,4,1,fichout);
                fwrite(&no_it,4,1,fichout2);
                fwrite(&nb,4,1,fichout2);
              )
            )
            if (flat==0)
              (
                if (no_lu>=K1)
                  (
                    flat=1;
                  )
                )
              )
            no_lu=lire_entier();
          )
        for (i=1;i<=20;i++)
          ( som2 += i*Amas[i]; )
        printf("\n* HISTOGRAMME DES n-CLIQUEES :\n");

```

```

for (i=1;i<=4;i++)
  ( printf("\n Nombre de %d-cliques = %d",i,cliq[i]); )
printf("\n\n --> som1 (Nbre d'activites totale) = %d\n", (int)som1);
printf("\n\n --> som2 (Nbre d'activites en amas) = %d\n", (int)som2);
printf("\n* HISTOGRAMME DES n-AMAS :\n");
for (i=1;i<=20;i++)
  ( printf("\n Nombre de %d-Amas = %d",i,Amas[i]); )
fclose(fichout);
fclose(fichout2);
fclose(fichin);
)

```

```

for (i=0;i<=50;i++)
  ( Amas[i]=0; )
for (i=0;i<=5;i++)
  ( cliq[i]=0; )
for (i=0;i<=150;i++)
  ( Tiso[i]=200;Tres[i]=200;Tamp[i]=200; )
flat=0;
no_lu=lire_entier();
while( (no_lu<=Kmax) && (feof(fichin)!=0) )
  ( nb=lire_entier();
    if (nb<=Ordre)
      for (i=0;i<=nb-1;i++)
        (
          neurone[i][0]=lire_entier();
          neurone[i][1]=lire_entier();
          /*sortie[neurone[i][0]][neurone[i][1]]=1;*/
        )
        /*imprime_mat();
        init_mat();*/
        if (flat==1) (les_amas());
        if (flat==0)
          (
            if (no_lu>=K1)
              (
                flat=1;
              )
            )
          )
        else
          (
            if (flat==1)
              (
                no_it=Ordre*(no_lu-1)+1;
                fwrite(&no_it,4,1,fichout);
                fwrite(&nb,4,1,fichout);
                fwrite(&no_it,4,1,fichout2);
                fwrite(&nb,4,1,fichout2);
              )
            )
            if (flat==0)
              (
                if (no_lu>=K1)
                  (
                    flat=1;
                  )
                )
              )
            no_lu=lire_entier();
          )
        for (i=1;i<=20;i++)
          ( som2 += i*Amas[i]; )
        printf("\n* HISTOGRAMME DES n-CLIQUEES :\n");

```



```

drapeau3=1;
break;
else (s++;)
}
if (drapeau3==0)
{
for (i=0;i<=5;i++)
{triplet[dim3][i]=ent2[i];}
Ntriplet[dim3]=1;
dim3+=1;
}
}
if (nb_un==4)
{
for (i=0;i<=7;i++)
ent2[i]=lire_entier();}
out=0;
drapeau4=0;
affect_T();
s=0;
while (s<dim4)
{
i=0;
while (i<8)
{Tq[quad[s][i]][quad[s][i+1]]=1;
i+=2;
}
}
for (i=0;i<=N-1;i++)
for (j=0;j<=N-1;j++)
{
if (T[i][j]!=Tq[i][j])
{out=1;break;}
}
}
if (out==0)
{
Ngq=1;
Nquad[s]+=1;
drapeau4=1;
init_T();
i=0;
while (i<8)
{Tq[quad[s][i]][quad[s][i+1]]=0;
i+=2;
}
break;
}
else
{
i=0;
while (i<8)
{Tq[quad[s][i]][quad[s][i+1]]=0;
i+=2;
}
s++;out=0;
}
} /*fin while(s<dim4) */
if (drapeau4==0)
{
for (i=0;i<=7;i++)

```

```

{quad[dim4][i]=ent2[i];}
Nquad[dim4]=1;
dim4+=1;
init_T();
}
}
if (nb_un>4)
{
for (i=0;i<=nb_un-1;i++)
{
k1=lire_entier();
k2=lire_entier();
}
}
Itlast=no_it;
no_it=lire_entier();
Nvide=no It-Itlast-1;
/*if (flat==0) (if (no_it>=K1) (initialisation();flat=1;))*/
}
}
fich_tr=fopen("tr_ama.ima", "w");
s=0;
while ( s<dim3 )
{
jj=0;
for (i=0;i<=2;i++)
{
neurone[i][0]=triplet[s][j];
neurone[i][1]=triplet[s][j+1];
jj+=2;
}
}
for (i=0;i<=2;i++)
for (j=0;j<=2;j++)
TT[i][j]=d_manhattan(neurone[i][0],neurone[i][1],neurone[j][0],neurone[j][1]);
}
}
for (i=0;i<=2;i++)
{
ii=0;
testi=0;
for (j=0;j<=2;j++)
{
if ( (TT[i][j])<=2) && (j!=i) )
{testi+=1;pos[ii]=j;ii++;}
}
if (testi==2)
{
fwrite (&neurone [pos [0]][0],4,1,fich_tr);
fwrite (&neurone [pos [0]][1],4,1,fich_tr);
fwrite (&neurone [i][0],4,1,fich_tr);
fwrite (&neurone [i][1],4,1,fich_tr);
fwrite (&neurone [pos [1]][0],4,1,fich_tr);
fwrite (&neurone [pos [1]][1],4,1,fich_tr);
fwrite (&Ntriplet [s],4,1,fich_tr);
break;
}
} /* fin for (i=0;i<=2;i++) */
s++;
} /* while ( s<dim3) */
x=44444;
fwrite (&x,4,1,fich_tr);

```

```

fclose(fich_tr);

fich_qu=fopen("qu_ama_ima", "w");
s=0;
while ( s<dim4 )
{
  jj=0;
  for (i=0; i<=3; i++)
  {
    neurone[i][0]=quad[s][jj];
    neurone[i][1]=quad[s][jj+1];
    jj+=2;
  }
  for(i=0; i<=3; i++)
  for (j=0; j<=3; j++)
  TT[i][j]=d_manhattan(neurone[i][0], neurone[i][1], neurone[j][0], neurone[j][1]);

  m=0;
  for(i=0; i<=3; i++)
  {
    ii=0;
    testi=0;
    for (j=0; j<=3; j++)
    {
      if ( (TT[i][j]<=2) && (j>i) )
      {testi+=i; pos[ii]=i;
      pos[ii+1]=j; ii+=2;
      }
    }
    ss=0;
    while(ss<testi)
    {
      for (j=0; j<ii; j++)
      {I[m]=pos[j]; m++;
      ss++;
      }
      /* for (i=0; i<=3; i++) */
    }
    ss=0;
    while(ss<m)
    {
      fwrite(sneurone[I[ss]][0], 4, 1, fich_qu);
      fwrite(sneurone[I[ss]][1], 4, 1, fich_qu);
      fwrite(sneurone[I[ss+1]][0], 4, 1, fich_qu);
      fwrite(sneurone[I[ss+1]][1], 4, 1, fich_qu);
      sst+=2;
    }
    s++;
  } /* while ( s<dim4 ) */
  x=5555;
  fwrite(&x, 4, 1, fich_qu);
  fclose(fich_qu);

printf( "\n\nNbtre triplets amas =%d\ntnt ( triplets repetees) = %d\n", dim3, Nt);
printf( "Nbtre quadruplets_amas =%d\tnq(quadruplets repetees) = %d\n", dim4, Nqq);
}

```

```

/*****
** VISU_CLIQ.C : lit le fichier "SPOT CLIQ.ENT"
** donne par le programme : amas.c et genere trois fichiers:
** - pour 2-cliques --> "pa.ima"
** - pour 3-cliques --> "tr.ima"
** - pour 4-cliques --> "qu.ima"
**
**
#include stdio
#include math
#define taille 6000

int N;
int Nvide;
int dim2, dim3, dim4;
/* La taille du reseau global
/* Nombre d'apparition de la configuration vide*/

int Jsgltn[128][128]; /* Potentiels de singleton */
int T[128][128], Tq[128][128];
int paire[taille][4]; /* vecteur paire[s]=(i1,j1,i2,j2) */
int Npaire[taille];
int Ntriolet[taille][6];
int qua[taille][8];
int Nqua[taille];
int ent[6], ent2[8];

static FILE *fich;
static FILE *fich_pa,*fich_tr,*fich_qu; /* pointeur du fichier de lecture */

/*****
** fonction lire entier :
** lit et retourne un entier du fichier de lecture
**
int lire_entier()
{
    int x;
    fread(&x,4,1,fich);
    return x;
}

/*****
** procedure initialisation :
** initialise les parametres globaux du programme
**
initialisation()
{
    int i,j;
    for(i=0;i<N-1;i++)
        for(j=0;j<N-1;j++)
            T[i][j]=0;
            Tq[i][j]=0;
}

for(j=0;j<taille-1;j++)
    Npaire[j]=0;
    Ntriolet[j]=0;

```

```

    Nqua[j]=0;
}
/*****
** affect_T()
{
    int s;
    s=0;
    while (s<=7)
        T[ent2[s]][ent2[s+1]]=1;
        s+=2;
}
/*****
** init_T()
{
    int s;
    s=0;
    while (s<=7)
        T[ent2[s]][ent2[s+1]]=0;
        s+=2;
}
/*****
** procedure stock_paire:
** stocke dans "pa.ima" les paires et leur potentiel
**
stock_paire()
{
    int i,s,x,nbpaire;
    nbpaire=0;
    fich_pa=fopen("pa.ima","w");
    s=0;
    while ( s<=dim2-1 )
        for (i=0;i<=3;i++)
            fwrite(&paire[s][i],4,1,fich_pa);
            nbpaire++;
            s++;
    printf("\nNbre 2-cliques (paires) = %d\n",nbpaire);
    x=3333;
    fwrite(&x,4,1,fich_pa);
    fclose(fich_pa);
}
/*****
** procedure stock triplet (en clique)
** stocke dans "tr.ima" les triplets
**
stock_triplet()
{
    int i,s,z,nbtrip,k;
    nbtrip=0;
    fich_tr=fopen("tr.ima","w");
    s=0;

```

```

while ( s<=dim3-1 )
{
    for (i=0;i<=5;i++)
        (fwrite(striplet[s][i],4,1,fich_tr);)
    fwrite(entriplet[s],4,1,fich_tr);
    nbtrip++;
    s++;
}
printf("\nNbre 3-cliques(triplets) = %d\n",nbtrip);
z=44444;
fwrite(x,4,1,fich_tr);
fclose(fich_tr);
}
/*-----*/
/*
 * Procedure stock quadruplets (en clique)
 * stocke dans "qu.ima" les quadruplets
 */
stock_quadruplet()
{
    int i,s,x,nbquad;
    nbquad=0;
    fich_qu=fopen("qu.ima","w");
    s=0;
    while ( s<=dim4-1 )
    {
        for (i=0;i<=7;i++)
        {
            fwrite(squa[s][i],4,1,fich_qu);
        }
        fwrite(snqua[s],4,1,fich_qu);
        nbquad++;
        s++;
    }
    printf("\nNbre 4-cliques(quadruplets) = %d\n",nbquad);
    z=55555;
    fwrite(x,4,1,fich_qu);
    fclose(fich_qu);
}
}
/*-----*/
/*
 * corps du programme principal
 */
main()
{
    char lect[40];
    int i,j,s,t,k;
    int il,jl,i2,j2;
    int itlast;
    int drapeau2,drapeau3,drapeau4;
    int nb_un,Kmax,n0_it,K1;
    int flat,outt,Ordre;
    int Nqg,Np,Nt;

    printf("\n\n
    ***SIMULATION D'UN RESEAU DE NEURONES***\n\n");
    printf("Lecture du fichier' SPTQ_CLIQ.ENT ' et reconstruit trois fichiers,\n");
    printf("sont resp.'PA.IMA'(paires en cliques), 'TR.IMA'(triplet en cliques),\n");
    printf("et 'QU.IMA'(quadruplets en cliques), generes par : 'VISU_CLIQ.C'\n");

    printf("\n\nEntrez la taille du reseau
    scanf("%d",&N);
    printf("%d x %d\n",N,N);
}
printf("Entrez le nombre total d'iterations ( Kmax ) ----> ");
scanf("%d",&Kmax);
printf("%d\n",Kmax);

printf("Ordre de reechantillonnage ( Ordre ) ----> ");
printf("%d\n",Ordre);

printf("Entrez le type de mode d'implementation du reseau :\n");
printf("Implementation Parallele --> Entrer , par '\n';
printf("Implementation Block-sequentiel --> Entrer , bseq '\n\n");
printf("
scanf("%s",flect);
printf("%s\n",lect);

itlast=1;t=0;flat=1;
outt=0;
Nqg=0;Np=0;Nt=0;
dim2=0;dim3=0;dim4=0;

initialisation();

fich=fopen("sptq_cliq.ent","r");
n0_it=lire_entier();
printf("n0_it = %d\n",n0_it);

while (feof(fich)==0)
{
    nb_un=lire_entier();
    if (nb_un==1)
    {
        il=lire_entier();
        jl=lire_entier();
        i2=lire_entier();
        j2=lire_entier();
        s=0;
        drapeau2=0;
        while (s<dim2)
        {
            if ( (paire[s][0]==il)&&(paire[s][1]==j1)&&(paire[s][2]==i2)
            &&(paire[s][3]==j2)) || ( (paire[s][0]==i2)&&(paire[s][1]==j2)&&(paire[s][2]==il)
            &&(paire[s][3]==j1)) )
            {
                Np++;
                Npaire[s]++;
                drapeau2++;
                break;
            }
            else (s++;)
        }
        if (drapeau2==0)
    }
}
}

```

```

    paire[dim2][0]=i1;
    paire[dim2][1]=j1;
    paire[dim2][2]=k2;
    Npaire[dim2]=1;
    dim2+=1;
  }
}
if (nb_un==3)
  for (i=0;i<=5;i++)
    ent[i]=lire_entier();
s=0;
drapeau3=0;
while (s<dim3)
  for (i=0;i<=5;i++) (test[i]=0;)
  for (i=0;i<=5;i++)
    {
      if (triolet[s][i]==ent(i))
        (test[0]+=1;)
      if (triolet[s][i]==ent((i+2)%6))
        (test[1]+=1;)
      if (triolet[s][i]==ent((i+4)%6))
        (test[2]+=1;)
    }
  if ( (triolet[s][0]==ent(0)) && (triolet[s][1]==ent(1))
    && (triolet[s][2]==ent(4)) && (triolet[s][3]==ent(5))
    && (triolet[s][4]==ent(2)) && (triolet[s][5]==ent(3)) )
    (test[3]=6;)
  if ( (triolet[s][0]==ent(4)) && (triolet[s][1]==ent(5))
    && (triolet[s][2]==ent(2)) && (triolet[s][3]==ent(3))
    && (triolet[s][4]==ent(0)) && (triolet[s][5]==ent(1)) )
    (test[4]=6;)
  if ( (triolet[s][0]==ent(2)) && (triolet[s][1]==ent(3))
    && (triolet[s][2]==ent(0)) && (triolet[s][3]==ent(1))
    && (triolet[s][4]==ent(4)) && (triolet[s][5]==ent(5)) )
    (test[5]=6;)
  if ( (test[0]==6) || (test[1]==6) || (test[2]==6) || (test[3]==6)
    || (test[4]==6) || (test[5]==6) )
    {
      Nt+=1;
      Ntriolet[s]+=1;
      drapeau3+=1;
      break;
    }
  else (s++;)
}
if (drapeau3==0)
  for (i=0;i<=5;i++)
    (triolet[dim3][i]=ent(i);)
  Ntriolet[dim3]=1;
  dim3+=1;
}
}
if (nb_un==4)
  for (i=0;i<=7;i++)
    ent2[i]=lire_entier();
  outt=0;
  drapeau4=0;
  affect_T();
  s=0;
  while (s<dim4-1)
    {
      i=0;
      while (i<=7)
        {
          Tq[qua[s][i]][qua[s][i+1]]=1;
          i+=2;
        }
      for (i=0;i<=N-1;i++)
        for (j=0;j<=N-1;j++)
          {
            if (T[i][j]!=Tq[i][j])
              (outt=1;)
          }
      if (outt==0)
        {
          Nqqt+=1;
          Nqua[s]+=1;
          drapeau4+=1;
          init_T();
          i=0;
          while (i<=7)
            {
              Tq[qua[s][i]][qua[s][i+1]]=0;
              i+=2;
            }
          break;
        }
      if (outt==1)
        {
          i=0;
          while (i<=7)
            {
              Tq[qua[s][i]][qua[s][i+1]]=0;
              i+=2;
            }
          s++;
          outt=0;
        }
    }
  /* fin while (s<dim4) */
  if (drapeau4==0)
    {
      for (i=0;i<=7;i++)
        {
          qua[dim4][i]=ent2[i];
          Nqua[dim4]=1;
          dim4+=1;
          init_T();
        }
    }
  Itlast=no_it;
  no_it=lire_entier();
  Nvide+=no_it-Itlast-1;
}
fclose(fich);

```

```
stock_paire();
stock_triplet();
stock_quaduplet();

printf( "\n*Np ( paires   repetees) = %d\n", Np);
printf( "**Nt ( triplet  repetees) = %d\n", Nt);
printf( "**Nq (quaduplets repetees) = %d\n", Nqq);
}
```

```

c      transcr2.for
      subroutine escrifor1(im,tab)
      integer i
      integer tab(10000),dim5
      character *40 fich
      fich='pa for.ima'
      open(1,file=fich,access='sequential',status='unknown',err=101,
      form='unformatted')
      dim5=im/5
      type *,im,dim5',im,dim5
      do i=1,dim5
      write(1)(tab((i-1)*5+k),k=1,5)
      end do
      goto 110
101  type *,'erreur ouverture 1'
110  close(1)
      return
      end

      subroutine escrifor2(im,tab)
      integer i,dim7
      integer tab(10000)
      character *40 fich
      fich='tr for.ima'
      open(2,file=fich,access='sequential',status='unknown',err=102,
      form='unformatted')
      dim7=im/7
      type *,im,dim7',im,dim7
      do i=1,dim7
      write(2)(tab((i-1)*7+k),k=1,7)
      end do
      goto 120
102  type *,'erreur ouverture 2'
120  close(2)
      return
      end

      subroutine escrifor3(im,tab)
      integer i,dim9
      integer tab(10000)
      character *40 fich
      fich='qu for.ima'
      open(3,file=fich,access='sequential',status='unknown',err=103,
      form='unformatted')
      dim9=im/9
      type *,im,dim9',im,dim9
      do i=1,dim9
      write(3)(tab((i-1)*9+k),k=1,9)
      end do
      goto 130
103  type *,'erreur ouverture 3'
130  close(3)
      return
      end

```

```
c      transcr3.for
      subroutine ecrifor2(im,tab)
      integer i
      integer tab(10000),dim5
      character *40 fich

      fich='qu_ama_for.ima'
      open(1,file=fich,access='sequential',status='unknown',err=101,
         form='unformatted')
      dim5=im/5
      type *,im,dim5',im,dim5
      do i=1,dim5
      write(1)(tab((i-1)*5+k),k=1,5)
      end do
      goto 110
101  type *,'erreur ouverture 1'
      close(1)
      return
      end

      subroutine ecrifor1(im,tab)
      integer i,dim7
      integer tab(10000)
      character *40 fich
      fich='tr_ama_for.ima'
      open(2,file=fich,access='sequential',status='unknown',err=102,
         form='unformatted')
      dim7=im/7
      type *,im,dim7',im,dim7
      do i=1,dim7
      write(2)(tab((i-1)*7+k),k=1,7)
      end do
      goto 120
102  type *,'erreur ouverture 2'
      close(2)
      return
      end
```


ANNEXE 3

*programme principal de l'algorithme neuronal
implémenté sur l'hypercube T-40*


```

if (stock_mat=="o")
{gere_eCr_matrices();}
keep_h();
printf("nbre de un total ds: %d \n", posabs, nb_un);
release_h();
} /* fin if (posabs < 16) */
close_l(ligne);
config_free(td_hyp);
break;
} /* fin case 's': */

case 't' :
{
init_config_tore_anneau_2();
lecture();
M=TAILLE/4;
N=TAILLE/8;
init_reseau();
/*init_fich_out();*/
nb_un=0;
j=1;
somt=0;
if (pos==0)
{
gettime(stdebut);
tdebut1=tdebut;
}
duree_com=0;
for (K=1; K<=KMAX; K++)
{
transfer();
/*if (lire_entree=='o') (gere_lect_fich_in(K));*/
proc_dessus();
proc_dessous();
proc_droite();
proc_gauche();
coins();
parcours_reseau();
if (duree<TINERTIE)
(duree+=1);
else
{duree=1;}
/* pour l'iteration parallele massive */
for (x=0; x<=M-1; x++)
for (y=0; y<=N-1; y++)
(memor[x+2][y+2]=sorties[x][y]);
/*gere_fich_out(K, cptr_un);*/
/*if (stock_entree=='o') (gere_eCr_fich_in(K, cptr_un_fibre));*/
if (pos==0)
{
nb_un +=cptr_un ;
}
/*close_fich_out();*/
/*if (stock_entree=='o')
(close_eCr_fich_in());
else
{if (lire_entree=='o') (close_lect_fich_in());}*/
if (pos==0) {gettime(efin);
printf("nbre de un total ds0 : %d \n", nb_un);
printf("la somme des temps : %.4f \n", somt);
ttot=(tfin-tdebut)*0.000064;
duree_com/=1000000.;
keep_h();
writestr_h(" Temps d'execution en secondes : ", STDOUT);
writer64_h(ttot, 4, STDOUT);
writeln_h(STDOUT);
}
}
}
}

writestr_h(" Temps de communication en secondes : ", STDOUT);
writer64_h(duree_com, 4, STDOUT);
writeln_h(STDOUT);
release_h();
}
} /* fin case 't' : */

close_l(lien);
config_free(td_tore);
break;
} /* fin case 't' : */

} /* fin switch() */
} /* fin main() */

```

BIBLIOGRAPHIE :

- M. A. Arbib (64), " Brains, Machines and Mathematics ", Mc Graw hill, (1964).
- H. Atlan (72), " L'Organisation Biologique et la Théorie de l'Information ", Hermann, (1972).
- H. Atlan, F. Fogelman, J. Salamon, G. Weisbuch (81), " Random Boolean Networks", Cybernetics and Systems, 12, pp 103-121, (1981).
- J. R. Barra (81), " Mathematical Basis of Statistics ", Academic press, (1981).
- A. Berlinet (86), " Estimation Fonctionnelle ", Cours de D.E.A, Grenoble, (1986).
- E. R. Berlekamp, J. H. Conway et R. K. Guy (82), " Winning Ways ", Academic press, (1982).
- J. Besag (74), " Spatial Interaction and Statistical Analysis of Lattice Systems ", J. Royal Statist.Soc., B 36, pp 192-254, (1974).
- J. P. Changeux(84), " L'Homme Neuronal ", Fayard, (1984).
- M. Cosnard(83), " Contribution à l'Etude du Comportement Itératif des Transformations Unidimensionnelles ", Thèse d'état, Grenoble I, (1983).
- P. Dallos (73), " The Auditory Periphery ", Academic press, New York, (1973).
- J. Demongeot (81), " Asymptotic Inference for Markov Random Fields on \mathbb{Z}^d ", Springer Series in Synergetics, 9, pp 254-267, (1981).
- J. Demongeot (87), " Random Automata Networks ", Automata Networks in Computer Science, F. Fogelman et al. (eds), Manchester University press, (1987).
- J. Demongeot (85), " Random Automata and Random Fields ", In Dynamical Systems and Cellular Automata, J. Demongeot et al. (eds), Academic Press, London, pp 99-110, (1984).
- J. Demongeot (81), " Etude Asymptotique d'un Processus de Contagion ", dans Biométrie et Epidémiologie, J. M. Legay et al. (eds), INRA, Paris, pp 52-143, (1981).
- J. Demongeot (83)a, " Coupling of Markov Process and Holley's Inequalities for Gibbs Measures ", in Proc. of The IXth Prague Conference, Academia, Prague, pp 9-18, (1983).
- J. Demongeot (83)b, " Systèmes Dynamiques et Champs Aléatoires : Application en Biologie Fondamentale ", Thèse d'état, Grenoble I, (1983).
- J. Demongeot (85), " Markov Processes and Population Entropy ", J. Math. Biol., (1985).
- J. Demongeot, M. Tchuente (87), " Dynamical Systems and Cellular Automata ", in Encyclopedia of Physical Science and Technology, Academic Press, New York, Vol 4, pp 464-469, (1987).
- J. Demongeot, J. Fricot (86), " Random Fields and Spatial Renewal Potentials ", in Disordered Systems and Biological Organization, E. Bienenstock et al. (eds), NATO ASI Series, Springer Verlag, New York, Vol F 20, (1986).

- E. E. Doberkat (81), " Stochastic Automata : Stability, Non Determinism and Prediction", Springer Verlag, (1981).
- J. L. Doob (53), " Stochastic Processes ", New York, Wiley, (1953).
- J. Durbin (76), " Kolmogorov-Smirnov Tests when Parameters are Estimated ", In Empirical Distributions and Processes, P. Gaussler and P. Revesz. Eds. Lecture Notes in Mathematics, N° 566, Berlin : Springer-Verlag, (1976).
- E. B. Dynkin (63), " Théories des Processus Markoviens ", DUNOD, Paris, (1963).
- M. J. Flynn (66), " Very High-Speed Computing System ", Proc. IEEE, 54, pp. 1901-1909, (1966).
- F. Fogelman (85), " Contribution à une Théorie du Calcul sur Réseaux ", Thèse d'état, Grenoble I, (1985).
- F. Fogelman, E. Goles, G. Weisbuch (82), " Specific Roles of The Different Boolean Mappings in Random Networks ", Bull. Math. Biol., Vol 44, N° 5, pp 715-730, (1982).
- F. Fogelman, E. Goles, G. Weisbuch (83), " Transient Length in Sequential Iteration of Threshold Functions ", Discrete Applied Math, 6, pp 95-98, (1983).
- F. Fogelman, E. Goles, D. Pellegrin (84), " Decreasing Energy Function as a Tool for Studying Threshold Networks ", Discrete Applied Math, (1984).
- F. Fogelman, P. Gallinari, Y. Le Cun, S. Thira (87), " Automata Networks and Artificial Intelligence ", In F. Fogelman and al. Eds. : Automata Networks In Computer Science. Manchester University Press, (1987).
- O. François (90), " Ergodicité des Processus Neuronaux ", CRAS, 310, pp 435-440, (1990).
- J. C. Frauenthal (80), " Mathematical Modeling in Epidemiology ", Springer Verlag, New York, (1980).
- [FPS] " Floating Point System Technical Publication Staff" T-series C and Fortran reference manual, release C001.
- F. R. Gantmacher (59), " Applications of The Theory of Matrices ", Interscience publisher, New York, (1959).
- D. Geman, S. Geman (84), " Stochastic Relaxation, Gibbs Distributions and Bayesian Restoration Of Images ", IEEE PAMI Transaction, 6, pp 721-741, (1984).
- S. Goldstein (81), " Entropy Increase in Dynamical Systems ", Israel Journal of Math, 38, pp 241-256, (1981).
- E. Goles, J. Olivos (80), " Periodic Behaviour of Generalized Threshold Functions ", Comm. Discrete Applied Math, 30, pp 187-189, (1980).
- E. Goles, J. Olivos (81), " Comportement Périodique des Fonctions à Seuil et Applications", Discrete Applied Math, 3, (1981).
- E. Goles (82), " Fixed Point Behaviour of Threshold Functions on a Finite Set ", Siam J. on discrete Math., 3, 4, (1982).

- E. Goles (83), " Dynamical Behaviour of Neural Networks ", R. R., IMAG, Grenoble, 386, (1983).
- E. Goles (85), " Comportement Dynamique de Réseau d'Automates ", Thèse d'état, Grenoble I, (1985).
- E. Goles, M. Tchuente (83), " Iterative Behaviour of Generalized Majority Functions ", Math. Soc. Sci., 4, pp 179-204, (1983).
- E. Goles, J. Olivos (81), " The Convergence of Symmetric Threshold Automata ", Information and Control, 51, N° 2, (1981).
- P. L. Gengoux, D. Trystram (89), " Comprendre l'Informatique Numérique ", édition Lavoisier, (1989).
- J. L. Gustafson, S. Hawkinson and K. Scott (86), " The Architecture of a Homogenous Vector Supercomputer ", J. of Par. and Distri. Computing, 3, pp 297-304, (1986).
- A. Herscovici (88), " Introduction Aux Grands Ordinateurs Scientifiques ", édition Eyrolles, (1988).
- D. O. Hebb (49), " The Organization of Behavior ", New York, Wiley, (1949).
- T. Hervé (87), " Champ Aléatoire et Information Neuronale : Application aux Voies Auditives", Thèse, INPG, Grenoble, (1987).
- T. Hervé, J. M. Dolmazon and J. Demongeot (90 a), " Random Field and Neural Information ", Proc. Natl. Acad. Sci., U.S.A., Vol. 87, pp 806-810, (1990).
- T. Hervé, T. Irino and H. Kawahara (90 b), " Representing Temporal Information in Auditory Periphery based on Random Field Theory ", Technical Report, H-90-40, Acoustical Soc. Japan, (1990).
- W. D. Hillis (82), " New Computer Architecture and Their Relationship to Physics or why Computer Science is no Good ", International Journal of Theoretical Physics, Vol. 21, N° 3-4, pp 255-262, (1982).
- W. D. Hillis (84), " The Connection Machine : A Computer Architecture Based on Cellular Automata ", Physica D, 10, pp 213-228, (1984).
- G. E. Hinton, J. A. Anderson (Ed.) (81), " Parallel Models of Associative Memory ", (1981).
- G. E. Hinton, T. Sejnowski, D. H. Ackley (84), " Boltzman Machines : Constraint Satisfaction Networks That Learn ", Carnegie Mellon University, Cognitive Science, 9, pp 147-169, (1984).
- J. J. Hopfield (82), " Neural Networks and Physical Systems with Emergent Collective Computational Abilities ", Proc. Nat. Acad. Sci. USA, 79, pp 2554-2558, (1982).
- J. J. Hopfield (84), " Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons ", Proc. Nat. Acad. Sci. USA, Vol 80, pp 3088-3092, (1984).
- J. J. Hopfield, D. W. Tank (85), " Neural Computation of Decisions in Optimization Problems ", Biological Cybernetics, Springer-Verlag, pp 141-152, (1985).

- B. A. Huberman, T. Hogg (84), " Adaptive and Self-Repair in Parallel Computing Systems ", Phy. Rev. Letters, Vol. 52, N° 12, pp 1048-1051, (1984).
- K. Hwang, F. Briggs (84), " Parallel Processing And Computer Architecture ", Mc Graw Hill, (1984).
- K. Patel Jagdish, C.H. Kapadia and D.B. Owen (76), " Handbook of Statistical Distributions ", Statistics : Textbooks and Monographs, Vol. 20, Marcel Dekker, Inc, (1976).
- N.L. Johson and S. Kotz (70), " Continuous Univariate Distributions ", Houghton Mifflin, Boston, Massachussets, Vol. 1 and 2, (1970).
- S. Karlin, H. M. Taylor (75), " A First Course in Stochastic Processes ", Academic Press, Inc, (1975).
- S. A. Kauffman (72), " The Organization of Cellular Genetic Control Systems ", In Lectures on Mathematics in the Life Science, Eds J. D. Cowan, Vol 3, pp 363-116, (1972).
- M. Khelladi, D. Benaouda (87), " Etude Statistiques d'un Processus Autorégressif d'Ordre Un Complexe ", Mémoire de D.E.A, Grenoble, (1987).
- S. Kirkpatrick, C. D. Gellat, M. P. Vecchi (83), " Optimization by Simulated Annealing", Science, 220, pp 671-680, (1983).
- T. Kohonen (84), " Self-Organization and Associative Memory ", Springer Verlag, (1984).
- K. Krickeberg (81), " Processus Ponctuel ", Ecole d'été de Saint Flour X, Lecture Note in Mathematics, 929, pp 205-313, (1981).
- H. T. Kung (82), " Why Systolic Architectures ? ", Computer, Vol. 15, pp 37-46, (1982).
- K. Kuppuswani, B. Tourancheau (88), " Evaluation The Performances On The [FPS] T-Series Hypercube Computer ", Research Report IMAG, 708 -I, Grenoble, (1988).
- B. Lacolle (84), " Sur Certaines Méthodes de la Physique Statistique ", Thèse d'état, Grenoble I, (1984).
- J.F. Lawless (82), " Statistical Models and Methods for Lifetime Data ", John willey & Sons, (1982).
- Y. Le Cun (87), " Modèles Connexionnistes de l'Apprentissage ", Thèse de Doctorat de l'Université de Paris VI, (1987).
- C. E. Leiserson (83), " Area Efficient VLSI Computation ", The MIT press, (1983).
- M. Loeve (62), " Probability Theory II, Graduate Texts in Mathematics ", Springer Verlag, New York, (1962).
- L. Martel & C. Soulier (88), " Simulation de Réseaux de Neurones sur Hypercube ", Rapport Projet 3ème année ENSIMAG, INPG, Grenoble, (1988).
- W. S. McCulloch, W. Pitts (43), " A Logical Calculus of The Ideas Immanent in Nervous Activity ", Bull. Math. Biophysics, 5, pp 115-133, (1943).

- R. Maynard & R. Rammal (81), " Ground State Structure of the Random Frustration Model in Two Dimensions ", In Numerical Methods in the Study of Critical Phenomena, Della Dora, Demongeot and Lacolle Ed., Springer Verlag, pp 104-116, (1981).
- M. Milgram (82), " Contribution aux Réseaux d'Automates ", thèse d'état, Compiègne, (1982).
- M. Minsky, S. Papert (69), " Perceptrons, An Introduction To Computational geometry ", The MIT Press, (1969).
- J. Neveu (77), " Processus Ponctuel ", Ecole d'été de Saint Flour VI, Lecture Note in Mathematics, 598, pp 300-447, (1977).
- A. Paz (71), " Introduction To Probabilistic Automata ", Academic press, New York, (1971).
- D. Pellegrin (86), " Algorithmes discrets et Réseaux d'Automates ", Thèse, Grenoble I, (1986).
- P. Peretto (84), " Collective Properties of Neural Networks, a Statistical Physics Approach", Biol. Cybernetics, 50, pp 51-62, (1984).
- D. A. Poplawski (88), " Mapping Rings And Grids Onto The FPS T-Series Hypercube Computer ", Parallel Computing, 7, pp. 1-10, (1988).
- J. C. Preston (74), " Gibbs States on Countable Sets ", Cambridge University Press, Cambridge, (1974).
- B. Prum (86), " Processus sur un Réseau et Mesure de Gibbs ", Edition Masson, (1986).
- P. Quinton, Y. Robert (89), " Algorithmes et Architectures Systoliques ", Edition Masson, (1989).
- P. Quinton (85), " Les Hyper - Ordinateurs ", La Recherche, N° 167, (1985).
- B. D. Ripley (88), " Statistical Inference for Spatial Processes ", Cambridge Un. Press, (1988).
- F. Robert (76), " Contraction en Norme Vectorielle, Convergence d'Itérations Chaotiques pour des Equations Non Linéaires de Point Fixe à Plusieurs Variables ", Linear Algebra and its Application, 13, pp 11-35, (1976).
- F. Robert (78), " Théorème de Perron Frobenius et Stein Rosenberg booléens ", Linear Algebra and its Application, 19, pp 237-250, (1978).
- F. Robert (86), " Discrete Iteration ", Springer Verlag, (1986).
- Y. Saad, M. H. Shultz (88), " Topological Properties Of Hypercubes ", IEEE Transactions on computers, Vol 37, N° 7, (1988).
- F. Spitzer (74), " Introduction Aux Processus de Markov à paramètres dans \mathbb{Z}^d ", Lecture Note in Mathematics, 390, pp 89-114, (1974).
- M. Tchuente (82), " Contribution à l'Etude des Méthodes de Calcul pour des Systèmes de Type Coopératif ", Thèse d'état, Grenoble, (1982).

- R. Thomas (79), Eds., " Kinetic Logic, A Boolean Approach to the Analysis of Complex Regulatory Systems ", Lecture Notes in Biomathematics, Vol 29, Springer-Verlag, (1979).
- D. Trystram (88), " Quelques Résultats de Complexité en Algorithmique Parallèle et Systolique ", Thèse INPG, (1988).
- [T-S] : T-Series, " Math Library Manual ", Release C.
- B. Tourancheau, G. Villard (88), " Manuel d'Utilisation De L'Hypercube FPS T-40 ", Polycopié INPG, (1988).
- S. D. Tuljapurkar (82), " Why Use Population Entropy ? It Determines The Rate of Convergence ", J. Math. Biol., 13, pp 325-345, (1982).
- A. M. Turing (36), " On Computable Numbers ", Proc. London Math. Soc., 2, pp 230-265, (1936).
- [UNIX], " Unix 4.2 BSD Kit De Base, Langage de Commande, Edition de Texte ", Courier et Donjon, Polycopié IMAG, (1983).
- G. Villard (88), " Calcul Formel et Parallèle : Résolution de Systèmes Linéaires ", Thèse INPG, (1988).
- J. Von Neumann (66), " Theory Of Self-Reproducing Automata ", Burks, A. W. ed., University of Illinois Press, Urbana Champaign, (1966).
- S. Wang, A. Schreiber, S. Rousset (89), " Connectionist Modelling of a Cognitive Model of Face Identification : Simulation of Context Effects ", Proceeding of IJCNN (International Joint Conference of Neural Networks), (1989).
- S. Wang (89), " Réseaux Multicouches de Neurones Artificiels ", Thèse INPG, (1989).
- B. Widrow, M. E. Hoff (60), " Adaptive Switching Circuits ", 1960 WESCON Convention Recod Part IV, 96-104, IRE, (1960).
- S. Wolfram (83), " Statistical Mechanics of Cellular Automata ", Review of modern physics, Vol. 55, pp 601-644, (1983).

Abstract :

This thesis consists in modeling of the neural networks located at downstream of the cochlear nucleus which constitutes the first layer of speech signals treatment coming from the peripheral auditory system, particularly from the basilar membrane. These signals represent the neuronal information carried by the auditory nerve fibers. The activity of these fibers is described through the recording of spike trains that constitutes the input to be treated by the network. The input is modeled by a two-dimensional binary random field (renewal stochastic point processes). This modeling takes into account the nature of the temporal, spatial and stochastic neuronal signal.

The general framework presented concerns the mathematical modeling of our neural network, the description of the parallel distributed memory machine "*Hypercube FPS T-40*" used as tool of the simulations, the implementation of the neural network on this parallel machine and finally the realization and interpretation of the simulation results. These topics are presented in four chapters as follows :

The first chapter concerns the general frame of the neural network, starting by the first models based on the threshold automata networks conceived by *W. S. McCulloch* and *W. Pitts* since 1943, cellular automata networks conceived by *J. Von Neumann* since 1948, etc...

The second chapter introduces the Gibbs measure, random fields and network models (deterministic and stochastic). Then, it presents the study on the ergodicity problem of the probabilistic neural networks. In the simple cases, our model of the neural network in parallel (resp. sequential) iteration mode is ergodic and converges to the unique invariant measure μ_P (resp. μ_S), in response to the stationary input.

The third chapter concerns the technical environment where the simulations of the neural network have been realized. It presents a general survey of the parallelization principles and an introduction to the parallel distributed memory machine "*Hypercube FPS T-40*".

Finally, the fourth chapter concerns the implementation of the neural network algorithm on the parallel machine "*Hypercube FPS T-40*", the numerical experimentations and the interpretation of the numerical results. The results are visualized graphically on a workstation *VINIX* (Image Processing) using the appropriate statistical measures which summarize the dynamical behaviour of the neural network.

Key words and phrases :

Stochastic Neural Networks, Automata Networks, Auditory System, Gibbs Measure, Random Fields, Ergodicity, Parallel Machine "*Hypercube FPS T-40*", *VINIX* Workstation.

