National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Outil logiciel pour le design et l'analyse d'expériences par les méthodes d'analyse factorielle et fractionnelle factorielle

Par
Yves do Régo

Thèse soumise
à l'Ecole des Etudes Supérieures et de la Recherche en vue de
l'obtention de la maîtrise ès sciences appliquées en informatique*

Département d'Informatique

Université d'Ottawa

Ottawa, Ontario

* Le programme de Maîtrise ès sciences appliquées en informatique est un programme conjoint avec l'Université de Carleton, administré par l'Institut d'Informatique d'Ottawa-Carleton.

ISBN 0-315-79996-X

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Je déclare être le seul auteur de cette thèse.

J'autorise l'Université d'Ottawa à prêter cette thèse à d'autres institutions ou individus, dans le but de recherches académiques.

De plus, j'autorise l'Université d'Ottawa de reproduire cette thèse par photocopies ou autrement, en sa totalité ou en parties, à la demande d'autres institutions ou individus, dans le but de recherches académiques.

Yves do Régo

à
mes grands parents
Claudio et Antoinette do Régo,
Félix et Marie Yéhoumè.

Remerciements

Avant d'entrer dans l'exposé à proprement dit de mes travaux, je voudrais à ce stade,

tout d'abord, avec déférence, remercier le Dr Tuncer Ören mon Directeur de thèse pour d'une part, m'avoir suggéré le présent sujet, et, d'autre part, m'avoir suivi dans mes recherches dont je soumets les résultats aux autorités académiques de l'Université.

Mes remerciements vont aussi à chacun des membres du groupe de recherche pour l'assurance de qualité en simulation et en génie logiciel de l'Université d'Ottawa en particulier au Dr Douglas King, pour son inestimable assistance.

Mes sentiments de filiale et fraternelle gratitude vont à

- mes parents Léopold et Bernardine do Régo,

- mes soeurs et frères Sylvie, Mireille et Jean-Claude pour leur compréhension et leurs grand soutien.

Finalement, je remercie Bell Canada pour le support financier qu'ils ont accordé au groupe de recherche pour l'assurance de la qualité en simulation et en génie logiciel du département d'informatique de l'Université d'Ottawa.

Résumé

Une expérimentation est conduite pour répondre à une où plusieurs questions que l'on peut se poser sur un sujet sous analyse. En vue de rendre systématique cette démarche, de nombreuses méthodes de construction de design d'expérimentation ont été conçues par des statisticiens; ces méthodes statistiques, en systématisant l'expérimentation, ont également rendu plus aisées l'analyse et l'interprétation de leurs résultats.

L'objectif de la présente thèse est le développement d'un logiciel appliquant les méthodes d'analyse factorielle et fractionnelle factorielle à un environnement de simulation sur ordinateur. Ces méthodes permettront la génération d'un ensemble de jeux d'essais qui une fois exécutés et analysés détermineront les effets et les implications que les paramètres d'entrées peuvent avoir sur le modèle de simulation sous étude.

Liste des Figures

Liste des Tables

Table des Matières

Introduction

Motivation

Un système peut être étudier de différentes manières soit par expérimentation directe, par la construction de prototypes, soit par définition de modèles mathématiques le représentant. En pratique, le coût et les implications du premier type d'expérimentation ont conduit au choix et développement de l'expérimentation à l'aide de modèles mathématiques (Korn, 1989). Un problème défini à l'aide d'un modèle mathématique peut avoir différents types de solutions, une solution réalisable, une solution satisfaisante ou une solution optimale. Ce modèle mathématique, est décrit le cas échéant, à l'aide d'une fonction analytique ou, à défaut, par recours à un modèle de simulation. Dans le cas de l'option pour un modèle de simulation, l'utilisateur ne pourra avoir qu'une solution satisfaisante à son problème étant donné que la solution analytique n'existe pas ou qu'elle implique des opérations non triviales. Par conséquent nous pouvons dire, à ce niveau, que la simulation d'un modèle représentant un objet "A" permet la description du comportement de cet objet dans diverses conditions.

Quel est l'intêret de la simulation d'un objet? Plusieurs réponses peuvent être apportées à cette question: l'objet "A" est soit inexistant ou trop coûteux de réalisation, soit l'exécution de certaines conditions sont impossibles sur le modèle réel. Par exemple, il est inconcevable que pour tester la résistance au feu d'une maison il faille la faire brûler. De même, en reprenant l'exemple de la maison d'habitation, il est inimaginable qu'il faille attendre que survienne un tremblement de terre pout étudier ses effets sur les matériaux et fondations de cette maison. En décrivant le comportement d'un objet "A" dans diverses conditions, la simulation permet d'apprendre, de reconnaître et, enfin de prévoir les réactions de l'objet "A" (Hunter et al, 1988).

La simulation est donc synonyme d'expérimentation à l'aide de modèles mathématiques (Korn, 1989). Dans le cas choisi, nous nous intéressons à la simulation sur ordinateur, plus précisement la simulation avec des modèles décrit à l'aide de langage de simulation sur des ordinateurs digitaux. Pour arriver à des conclusions valables à partir d'un modèle sous étude, celui-ci doit être exécuté sous diverses conditions, ceci est également vrai quand il s'agit de modéle simulé sur ordinateur. Aussi, un bon environnement de simulation devra-t-il afficher, enregistrer les résultats et interagir avec l'utilisateur de manière à lui permettre de modifier le modèle ou certains paramètres pour une autre exécution. Ainsi, pour pouvoir se concentrer sur ses expériences, l'utilisateur ou l'expérimentateur ne devra pas être confronté à des problèmes concernant par exemple la programmation du modèle.

Une expérience étant au sens général définie comme "une suite d'actions effectuées en vue de répondre à une ou plusieurs questions que l'on peut se poser sur un sujet (Law et Kelton, 1991), nous pouvons dire que l'expérimentation est l'ultime source permettant d'acquérir de nouvelles connaissances, de nouvelles lois et de nouvelles applications de ces dernières (Connor et Zelen, 1959). L'expérimentation est un art qui dépend des connaissances préalables de l'expérimentateur. En effet, ce dernier ne pourra déduire de nouvelles lois qu'en se basant sur celles préalablement acquises. Comme tout oeuvre humaine, il y a du bon travail et du moins bon. Dans le cas sur lequel nous nous penchons, les bons produits seront différenciés des moins bons en fonction du temps que l'expérimentateur devra consacrer à l'analyse et à l'interprétation des résultats de chacune des expériences. L'expérimentateur comme nous l'avons souligné précédemment devra être libéré des considérations concernant la programmation du modèle à étudier et d'autres problèmes liés à celle-ci. Mais, la liberté de l'expérimentateur devra être plus ou moins limité au niveau du design de ses expériences car de celui-ci dépend l'analyse des données qui peut être soit simple, soit complexe.

Objectifs de la thèse

L'objectif de cette thèse est l'implantation d'une méthode statistique permettant le design et l'analyse d'expériences conduites sur des modèles de simulation. Cette implantation se traduira par le développement d'un environnement de simulation et d'expérimentations sur ordinateur. Le noyau de cet environnement logiciel sera un générateur d'expériences ou de jeux d'essais respectant les techniques des méthodes d'analyse factorielle et fractionnelle factorielle. Alors que la méthode factorielle permettra la génération de toutes les combinaisons de jeux d'essais pouvant être exécutées sur le modèle sous étude, la méthode fractionnelle s'intèressera beaucoup plus à la recherche d'un sous-ensemble d'expériences qui permettra une analyse tout aussi bonne sans pertes préjudiciables d'informations. Par ailleurs le design ou la génération d'expériences étant le plus souvent inséparables de l'analyse des données, nous présenterons les méthodes d'analyses permettant de détecter non seulement les conditions générant une solution optimale au niveau des points sous études, mais, aussi les effets des paramètres de décisions dans le contexte de l'exécution du modèle. Les différentes analyses statistique faites sur des modèles de simulation ne sont possible qu'après la récupération de l'indice de performance (la réponse du système). Cet indice au niveau de notre logiciel devra être défini de façon explicite par l'utilisateur avant d'effectuer son expérimentation. Le type de cette indice devra être obligatoirement numérique.

Notre environnement logiciel permettra l'intégration des opérations de design, d'expérimentation et d'analyse statistique de modèle de simulation, opérations qui la plupart du temps sont réalisées par des logiciels différents. Nous pouvons citer par exemple NEMS un environnement pour l'expérimentation numérique (Birta et So, 1990), TurboSim, un logiciel permettant l'automatisation de l'analyse statistique de simulation à évènements discrets (Mourant, 1986) et le système expert développé par Jávor (1986).

Pour rendre compte des résultats de notre thèse, nous présenterons successivement:

- Dans le chapitre 1 quelques définitions de base, suivis de la description de différentes méthodes d'expérimentation qui feront partie d'une table comparative, qui nous a permis de justifier le choix des méthodes factorielles et fractionnelles factorielles implantées dans notre logiciel et développées dans les deux chapitres suivants.
- La méthode d'analyse factorielle. (Chapitre 2)
- La méthode d'analyse fractionnelle factorielle. (Chapitre 3)
- Le concept des réseaux de Pétri suivi d'une présentation du logiciel Voltaire. (Chapitre 4)
- Le logiciel SimAd qui a été développé en vue de conduire des expérimentations basées sur la méthode du design factoriel et celle du design fractionnel factoriel. (Chapitre 5)
- Le résultat expérimental d'un modèle de simulation exécutés avec SimAd. (Chapitre 6)
- et, pour terminer les résultats de notre thèse ainsi que les développements futurs du logiciel SimAd qui pourraient être poursuivis.

Chapitre 1 - Différentes méthodes d'expérimentation

Dans ce chapitre après avoir donné des définitions de base, nous présenterons différentes méthodes d'expérimentation telle que celle du design aléatoire, du design aléatoire par blocs, du carré latin et celle du carré de Youden. Ces méthodes vont être inclues dans une table comparative qui nous a permis de sélectionner les designs qui sont implantés dans notre logiciel.

1.1 Définitions de base

La vie de l'Homme est faite d'expériences. Ainsi, dès mon plus jeune âge, j'ai à l'instar de bien de jeunes enfants après avoir joué avec le feu et m'être brûlé, su par "expérimentation" les effets que pouvaient avoir les flammes sur un doigt. Ainsi, même si je ne cherchais pas à répondre au problème doigt/feu, je peux utiliser la définition suivante:

Définition 1.1: *Une expérimentation* est un ensemble d'expériences effectuées dans le but de répondre à une ou plusieurs questions que l'on peut se poser sur un sujet sous étude.

Toutefois, avant d'effectuer son expérimentation, l'analyste ou l'expérimentateur doit connaître l'objet sur lequel ses expériences seront conduites. En d'autres termes "l'unité expérimentale" doit être connue.

Définition 1.2: L'*unité expérimentale* est l'objet sur lequel sera appliqué les expériences et sur lequel on mesurera et analysera les données sous études.

Dans la plupart des tests d'agriculture, par exemple, l'unité expérimentale est généralement une partie du champ, plutôt qu'une plante. Une fois l'unité expérimentale déterminée, l'expérimentateur devra identifier les facteurs ou paramètres de décisions qu'il désire étudier; ceux-ci seront un sous-ensemble des variables du modèle sous étude.

Définition 1.3: Les variables indépendantes ayant des effets positifs ou négatifs sur la réponse d'un système sont appelées *facteurs.* Ces facteurs pourront avoir plusieurs niveaux qui seront constitués par les différentes valeurs qu'ils peuvent prendre au début de chaque expériences (Mendenhall et Sincich, 1984, p. 544).

Quand toutes ces données seront réunies, l'expérimentation pourra commencer. Supposons que nous voulons comparer la taille moyenne de deux groupes d'étudiants provenant de deux universités différentes, il est évident que nous ne choisirons pas des étudiants appartenant à l'équipe de basket-ball d'une université et ceux de l'équipe de gymnastique de l'autre. Si nous voulons prouver ou valider les résultats obtenus à la suite de cette comparaison, nous devons choisir de façon aléatoire les étudiants dans chacune des universités. Par conséquent pour assurer la qualité d'une expérimentation il faut prendre la précaution de prélever ou d'attribuer de façon aléatoire les niveaux des facteurs et/ou les sujets sous étude. En effet dans une expérimentation, le choix aléatoire et la répétition des expériences permettent de valider les résultats obtenus.

Définition 1.4: La *reproduction* est la répétition d'une expérience ou d'un traitement dans le but d'augmenter ou de valider la précision des résultats obtenus.

La répétition des expériences est le moyen de déceler les effets découlant de l'environnement dans lequel celles-ci sont faites. Ces répétitions permettront d'identifier et de distinguer les effets ou variations aléatoires de ceux ou celles qui sont systématiques (Hunter et al, 1988).

1.2 Le design aléatoire

Le design aléatoire est une méthode appropriée quand nous avons N (nombre entier) unités expérimentales pour l'expérimentation et k traitements ou différents niveaux des paramètres d'entrées à contrôler. Pour illustrer cette méthode, nous donnerons l'exemple suivant: un analyste veut étudier les effets de trois méthodes de conditionnement sur la résistance à la cassure d'un métal donné. Il dispose d'un lot de N=15 échantillons de ce métal auxquels il applique les trois méthodes de conditionnement. Ceci est un exemple typique d'une expérimentation aléatoire à 1-facteur, car, nous n'avons qu'un seul facteur expérimental à étudier, la méthode de conditionnnement. Etant donné que nous avons trois méthodes de conditionnement, le nombre de traitements k de cet exemple est égal à 3. Le nombre d'unité expérimentale n auquel nous assignerons aléatoirement les méthodes, est égal à 5. A la fin de l'expérimentation on a les données figurant dans la table 1.1 (Hunter et al, 1988).

Table 1.1 Force de résistance T du métal, gr/mm^2

	Méthode 1	Méthode 2	Méthode 3
observations	553 550 568 541 537	553 599 579 545 540	492 530 528 510 571
Total T	2749	2816	2631
n	5	5	5
Moyenne y	549,8	563,2	526,2
Estimation de la variance	145,7	626,2	864,2
Degrés de liberté	4	4	4

L'étude des données de la table 1.1 peut être effectuer par une technique statistique de base, l'analyse de la variance développée ci-après. Cette analyse permet d'analyser les données issues d'un design aléatoire en partitionnant la somme des carrés SS(Total) en deux sommes des carrés SSE et SST (Figure 1.1) représentant respectivement la somme des carrés des erreurs et la somme des carrés des traitements.

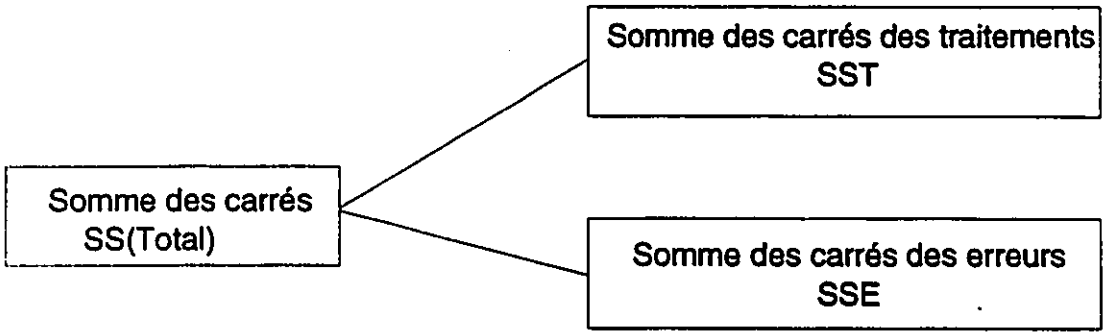

Figure 1.1: Partition de la somme des carrés total SS(Total) pour un design aléatoire (Mendenhall et Sincich, 1984, p. 549)

L'analyse de la variance de la Table 1.1 est dor · ci-dessous

Etant donnés T la somme des totaux des méthodes 8196
N le nombre total d'observations 15
n_i le nombre d'observations par méthodes 5 et
i le nombre de méthodes 1,2,3

Le total de la somme des carrés avant correction est:

$$\sum_{i=1}^{3} y^2 = 4.488.348$$

CM = facteur de correction:

$$CM = \frac{T^2}{N} = \frac{8196^2}{15} = 4\,478\,294{,}4$$

SS(Total) = Somme des carrés après correction à (N-1) degrés de liberté (DF):

$$\sum_{i=1}^{3} y^2 - CM = 4\,488\,348 - 4\,478\,294{,}4 = 10\,053{,}6$$

SST = Somme des carrés des traitements à (n_i-1) degrés de liberté:

$$\sum_{i=1}^{3} \frac{T_i^2}{n_i} - CM = \frac{22\,409\,018}{5} - CM$$

$$= 4481803{,}6 - 4478294{,}4 = 3509{,}2$$

SSE = Somme des carrés pour l'erreur à (N-n_i) degrés de liberté:
= SS(Total) - SST
= 10 053,6 - 3509,2 = 6544,4

Les moyennes des carrés se calculent des manières suivantes:

Moyenne des carrés des traitements $MST = \frac{SST}{n_i - 1}$

Moyenne des carrés pour l'erreur $MSE = \frac{SSE}{N - n_i}$

La page suivante présente les formules de calculs nécessaires à l'analyse de la variance pour un design aléatoire.(Mendenhall et Sincich, 1984, p. 550)

Somme de toutes les N mesures = $\sum_{i=1}^{k} y_i$

Moyenne de toutes les N mesures = $\bar{y}$

Somme des carrés de toutes les n mesures = $\sum_{i=1}^{k} y_i^2$

Correction de la moyenne = CM

$$= \frac{(\text{Total des observations})^2}{\text{Nombre total d'observations}}$$

$$= \frac{\left(\sum_{i=1}^{k} y_i\right)^2}{k}$$

SS(Total) = Somme des carrés des totaux

= (Somme des carrés de toutes les observations) - CM

$$= \sum_{i=1}^{n} y_i^2 - CM$$

SST = Somme des carrés des traitements

$$= \left(\begin{array}{l}\text{Somme des carrés des traitements avec}\\ \text{chaque carré divisé par le nombre}\\ \text{d'observations pour le traitement}\end{array}\right) - CM$$

$$= \frac{T_1^2}{n_1} + \frac{T_2^2}{n_2} + \ldots + \frac{T_k^2}{n_k} - CM$$

SSE = Somme des carrés pour l'erreur

= SS(Total) - SST

MST = Moyenne des carrés pour les traitements

$$= \frac{SST}{k-1}$$

MSE = Moyenne des carrés pour l'erreur

$$= \frac{SSE}{N-k}$$

Figure 1.2 Formules pour l'analyse de la variance d'un design aléatoire

Dans l'exemple précédent les n_i sont tous égaux. Les designs qui ont un nombre d'observations égale dans chaque traitement sont les plus utilisés parce qu'ils permettent de faire des comparaisons sans restrictions entre les traitements. Le design aléatoire est simple à organiser et à analyser. Il apparait, comme le meilleur des designs pour une expérimentation quand toutes les données sont homogènes et l'environnement est bien controlé pendant les expériences.

En conclusion, nous pouvons dire que la force de ces designs réside dans la simplicité de leur construtions, de leur analyse et aussi dans leur flexibilité en terme de nombre d'expériences et de traitements assigné aux unités expérimentales. La table 1.2 présente le format général de présentation des résultats provenant d'un design aléatoire.

Table 1.2 Design aléatoire

	Traitements				
Expériences	1	2	3	. . .	k
1	y11	y12	y13	. . .	y1k
2	y21	y22	y23	. . .	y2k
3	y31	y32	y33	. . .	y3k
:	:	:	:		:
:	:	:	:		:
n	yn1	yn2	yn3	. . .	ynk

Le design aléatoire est le nom donné au "design à 1-facteur" uniquement pour le distinguer des autres designs d'expérimentation comprenant dans leur structure la notion de répétition et/ou le principe de blocs.

1.3 Le design aléatoire par blocs

Le design expérimental que nous avons présenté dans la section précédente est applicable uniquement dans une situation d'homogénéité dans les unités. Dans la cas d'hétérogéinité dans les unités nous aurons recours au design aléatoire par blocs dont nous illustrerons la méthode par l'application suivante: on désire comparer le temps pris pour effectuer des traitements banquiers journalier par trois différents programmes. Dans un design aléatoire cette expérimentation peut-être effectuer en sélectionnant 15 jours et en attribuant de façon aléatoire aux trois programmes les reçus correspondant à 5 jours de travail. Cette méthode donnerait des résultats moins précis que si on utilisait des reçus correspondant à 5 jours et que durant chaque jour on exécute chacun des trois programmes. Cette méthode est plus précise parce qu'elle prend en considération le fait que le temps nécessaire au traitement des reçus dépend du niveau d'activité quotidien et de la complexité de la transaction. En faisant les comparaisons au jour le jour nous éliminons les variations pouvant exister entre des jours différents. Ce design aléatoire par blocs est représenté dans la figure 1.3 (Mendenhall et Sincich, 1984, p. 561).

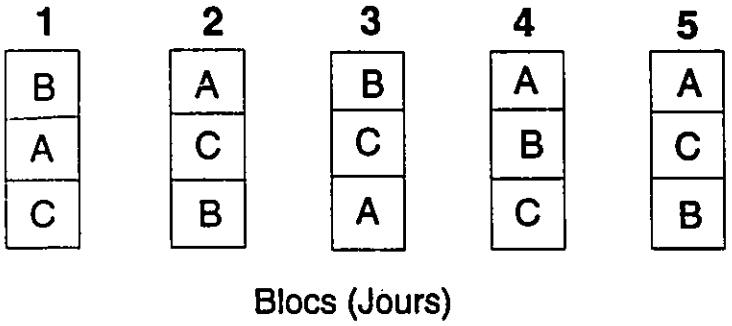

Figure 1.3 : Diagramme pour un design aléatoire par bloc contenant 5 blocs et trois traitements

Définition 1.6: Un *design aléatoire par blocs* permet la comparaison de N traitements évoluant dans b blocs, chacun de ces blocs contient exactement p unités expérimentales. Les traitements sont attribués aux unités de chaque bloc de façon aléatoire. La partition de la somme des carrés total pour un *design aléatoire par blocs* est représentée dans la figure 1.4.

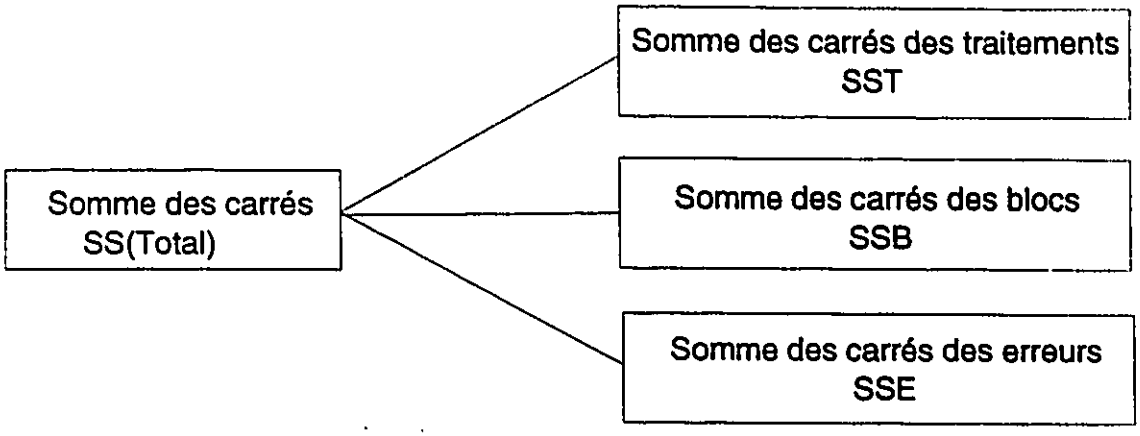

Figure 1.4: Partition de la somme des carrés pour un design aléatoire par bloc (Mendenhall et Sincich, 1984, p. 562)

Le calcul de SST et SSB est le même que celui du calcul de SST pour un design aléatoire. Une fois ces données obtenues SSE est trouvé en effectuant une soustraction:

$$SSE = SS(Total) - SST - SSB$$

La moyenne des carrés des traitements est:

$$MST = \frac{SST}{p - 1}$$

La moyenne des carrés des blocs est:

$$MSB = \frac{SSB}{b - 1}$$

La moyenne des carrés des erreurs est:

$$MSE = \frac{SSE}{N - p - b + 1}$$

1.4 Le design du carré latin

Ce design est utilisé pour étudier les effets de différents niveaux d'un facteur quand il y a deux sources de variations ou de non-homogénéité dans les conditions affectant les résultats des expériences. Ces sources de non-homogénéité peuvent être par exemple en agriculture les deux directions d'un champ, ou dans l'environnement d'une industrie, les machines et leurs positions. Ces sources de non-homogéinité sont appelées en statistique les "variables de blocage". Dans ces designs la variable étudiée et le traitement expérimental sont associés aux variables de blocages d'une maniére prédéfinie. Pour utiliser le *"carré latin"* l'expérimentateur devra respecter deux conditions:

- 1. Les nombre de lignes, colonnes et traitements doivent être égaux.
- 2. Il ne doit pas avoir d'intéractions entre les lignes, les colonnes, et le facteur étudié (Hunter et al., 1988).

Supposons que nous voulons comparer l'efficacité de quatre tissus à la résistance de l'air et que nous disposons pour cela d'une machine pouvant tester quatre tissus en même temps. Les deux variables de blocage, dans ce cas, pourraient être les variations d'une exécution à une autre et les variations entre les positions de la soufflerie. Pour cet exemple nous aurons la table 1.3 se trouvant à la page suivante:

Table 1.3 Un carré latin de 4 X 4

	Numéro de Position			
Exécution	1	2	3	4
1	C1	C2	C3	C4
2	C2	C3	C4	C1
3	C3	C4	C1	C2
4	C4	C1	C2	C3

Dans le cas d'un carré latin de 4 X 4 nous avons quatre carrés différents. Un des quatres peut être choisi de façon aléatoire et la procédure par la suite est la suivante:

- 1. Permuter les colonnes de façon aléatoire.
- 2. Permuter les lignes de façon aléatoire.
- 3. Affecter aléatoirement les lettres aux traitements.

Les résultats d'un *carré latin* peuvent être enregistrés dans une table à deux dimensions similaire au plan lui-même. L'analyse de ce design suppose qu'il n'existe aucune intéraction entre les lignes, colonnes et traitements. Si une ou plusieurs interactions existent les estimations issues de l'analyse ne pourront être que biaisées. Les difficultés que nous pouvons rencontrer au niveau de l'analyse d'un carré latin peuvent être évitées par l'utilisation de répétitions partielles du design comme dans le cas du *carré de Youden.*

1.5 Le design du carré de Youden

Le *carré de Youden* permet comme le *carré Latin* deux sources de variations non homogènes, mais contrairement à ce dernier ses conditions d'utilisation sont moins restrictives. En effet, dans le *carré Latin* comme nous l'avons dit précédemment une restriction existe au niveau du nombre de lignes, colonnes et traitements qui doivent être égaux. Le *carré de Youden* quant à lui doit avoir le même nombre de colonnes et de traitements, mais peut avoir un nombre quelconque de lignes. Mais bien qu'elle soit plus facile que celle du *carré latin*, l'analyse du *carré de Youden* doit être effectuée avec le plus grand soin. En particulier au niveau de la moyenne des traitements qui doit être ajustées à chaque ligne avant que toute comparaison soit possible (Nombre de lignes quelconques) (Hunter et al, 1988).

1.6 Comparaison de différents types de design d'expérimentation

Pourquoi avons nous choisi d'utiliser ces designs? D'une part parce qu'ils offrent certains avantages en termes d'économie d'expériences à réaliser et d'autre part parce qu'ils permettent une obtention assez aisée des estimations des effets, des estimations valide de la variance et d'autres informations. Les designs d'expérimentations peuvent être classifiés en tenant compte de plusieurs critères tel que:

- le nombre de facteurs sous études (un-facteur ou plusieurs facteurs)
- la structure de la méthode d'expérimentation (factoriel, avec blocage, etc)
- le type d'information recherché (estimations des effets, estimation de la variance).

Nous présentons la table 1.4 qui nous a permis de sélectionner la méthode factorielle nos critères de sélection étaient bien entendus le ou les designs répondant à notre objectif tout en offrant le moins de restrictions au niveau de leur structure et le maximum d'informations après analyse. Cette table est tirée de Hunter, J. S. et al. (1988).

Table 1.4 Comparaison de design d'expérimentation

Design	Type d'application	Structure	Informations
Aléatoire	Approprié quand un seul facteur est sous étude.	**Bases:** Un facteur est analysé par allocation aléatoire d'unités expérimentale au niveau des facteurs **Blocage:** aucun.	1. Estimation et comparaison des traitements. 2. Estimation de la variance.
Aléatoire par blocs	Approprié quand un facteur est analysé et que l'environnement peut être divisé en blocs ou groupes homogènes.	**Bases:** Chaque traitement ou niveau de facteur est exécuté dans chaque bloc. **Blocage:** Souvent par rapport à une variable ou facteur.	1. Estimation et comparaison des effets des traitements non bloqués. 2. Estimation de l'effet des blocs. 3. Estimation de la variance.

(suite à la page suivante)

Design	Type d'application	Structure	Informations
Carré Latin	Approprié quand un facteur est sous étude et que les résultats peuvent être affectés par deux autres facteurs ou deux sources non-homogènes.	**Bases:** Voir section 1.4.	1. Estimation et comparaison des effets des facteurs sans relation avec les deux variables de blocage. 2. Estimation et comparaison des effets des deux variables de blocage. 3. Estimation de la variance.
Carré de Youden	Même que le carré Latin mais les nombres de colonne, ligne et traitement ne doivent pas obligatoirement être égaux.	**Bases:** Chaque traitement apparait une seul fois dans chaque ligne. Le nombre de traitement doit être égale au nombre de colonnes. **Blocage:** Par rapport au autres facteurs (dans les deux sens).	1. Même que le carré Latin.

(suite à la page suivante)

Design	Type d'application	Structure	Informations
Factoriel	Approprié quand plusieurs facteurs sont sous étude et quand les interactions sont importantes.	**Bases:** Plusieurs facteurs sont analysés à plusieurs niveaux en exécutant toutes les combinaisons de facteurs et niveaux. **Blocage:** aucun.	1. Estimation et comparaison des effets de plusieurs facteurs. 2. Estimation possible des effets des interactions. 3. Estimation de la variance.
Fractionel factoriel	Approprié quand il y a plusieurs facteurs et qu'il est pratiquement impossible d'exécuter toutes les combinaisons de facteurs et niveaux.	**Bases:** Plusieurs facteurs sont analysés à différents niveaux mais seulement un sous-ensemble des combinaisons du factoriel complet sont exécutées. **Blocage:** Possible.	Même que factoriel à l'exception de la variance pour certaines fractions.

Chapitre 2 - La méthode d'expérimentation factorielle

Dans le chapitre précédent, nous avons présenté différentes méthodes d'expérimentations appropriées pour des études ne faisant intervenir qu'une seule variable (facteur) ou pour des modèles assurant l'inexistence d'interactions entre les variables. Ces méthodes n'ont pas été choisies pour le développement de notre logiciel parce qu'elles ne répondent pas à l'objectif que nous nous sommes fixé. Cet objectif nous le rappelons est le développement d'un logiciel permettant l'analyse des effets des paramètres sur la réponse du modèle de simulation. Pour ce faire nous présentons dans ce chapitre et le suivant des méthodes qui malgré leurs limites se sont imposées par la simplicité de leur design et celle de leur analyse.

2.1 Le design factoriel

La différence entre le design factoriel et ceux présentés précédemment réside surtout dans le fait que ce design permet de gérer plusieurs facteurs qui peuvent être initialisés à différents niveaux. Par facteur, nous rappelons que nous entendons variables, paramètres d'entrées ou paramètres de décisions et par niveau nous entendons la valeur que ces facteurs reçoivent au début de chaque expérience. Ces facteurs représentant des données réelles pourront être de deux types: qualitatif et quantitatif. Dans l'exemple qui suit, nous utiliserons deux facteurs quantitatifs. Si l'on suppose qu'une expérience fait intervenir un temps d'attente et la longueur d'une file d'attente nous pouvons avoir les valeurs suivantes faisant office de niveaux: 10 et 15 secondes pour le temps d'attente, et de 4 à 7 éléments pour la longueur de la file.

Etant données les notations suivantes:

temps d'attente = t, $t_0 = 10$, $t_1 = 15$.

file d'attente = f, $f_0 = 4$, $t_1 = 7$.

Le nombre d'expériences d'un design factoriel est fonction du nombre de variables à étudier mais aussi du nombre de niveau que chaque variable possède. La fonction générant ce nombre est n^v où n représente le nombre de niveaux des variables et v le nombre de variables.

Dans notre cas nous aurons donc 2^2 combinaisons de traitements ou expériences possible:

- $t_1 f_1$, les deux facteurs à leur niveau le plus haut,
- $t_1 f_0$ et $t_0 f_1$ un facteur au niveau haut et l'autre au niveau bas,

et $t_0 f_0$, les deux facteurs à leur niveau bas.

Pour exécuter un design factoriel général, l'expérimentateur devra sélectionner un nombre fixe de niveaux pour chacun des facteurs (variables) et faire chacune de ses expériences en se basant sur les combinaisons possibles. S'il y a l_1 niveaux pour la première variable, l_2 pour la seconde, ..., et l_k pour la k ième variable, l'arrangement complet sera le suivant:

$l_1 \times l_2 \times \ldots \times l_k$ expériences.

cet arrangement est appelé un design factoriel de $l_1 \times l_2 \times \ldots \times l_k$ expériences. Par exemple, un design factoriel de 2 x 3 x 5 demanderait 2 x 3 x 5 = 30 expériences, et dans notre cas $2 \times 2 = 2^2$ expériences. Cette thèse ne s'interressera qu'aux designs factoriels ne faisant intervenir que deux niveaux par facteurs.

2.2 Notations et réprésentation

Il existe plusieurs types de représentation d'un design factoriel à deux niveaux. L'une d'elle identifie le niveau le plus haut de chaque facteur par soit la première lettre de chaque facteur au niveau le plus haut ou par l'utilisation de 0 et de 1 ou encore par les signes "-" et "+". Une fois le code choisi, un design factoriel est généralement représenté à l'aide d'une table à deux dimensions. Le nombre de facteurs du design déterminera le nombre de colonnes de la table et son nombre de lignes sera égale à $2^{(\text{nombre de facteurs})}$. Cette table appelée la matrice de design contiendra les valeurs qu'auront les facteurs pour chacune des expériences. Ces valeurs seront créées en respectant l'ordre standard ou ordre de Yates (Box, Hunter et Hunter, 1978). Les différents types de représentations et notations sont présentés dans la table 2.1. (Box, Hunter et Hunter, 1978)

Table 2.1 Notations possibles d'un design factoriel de 2^3

Exécution	t c k		t c k
1	- - -	1	0 0 0
2	+ - -	t	1 0 0
3	- + -	c	0 1 0
4	+ + -	tc	1 1 0
5	- - +	k	0 0 1
6	+ - +	tk	1 0 1
7	- + +	ck	0 1 1
8	+ + +	tck	1 1 1

Nous utiliserons un design factoriel de 2^3 expériences pour illustrer cette méthode. Ce design permettra l'analyse des effets qu'auront deux variables quantitatives (température et concentration) et une variable qualitative (catalyseur) sur la production d'un réacteur chimique. Les données constituant ce design sont représentées dans les tables 2.2.a et 2.2.b (Box, Hunter et Hunter, 1978, p. 308).

Table 2.2.a Données d'un design factoriel de 2^3 expériences
(valeurs réelles des variables)

Expériences	température (degré) T	concentration (%) C	catalyseur (A ou B) K	réponse (grammes) y
1	160	20	A	62
2	180	20	A	72
3	160	40	A	54
4	180	40	A	68
5	160	20	B	52
6	180	20	B	83
7	160	40	B	45
8	180	40	B	80

Table 2.2.b Données d'un design factoriel de 2^3 expériences
(valeurs codées des variables)

Expériences	température (degré) T	concentration (%) C	catalyseur (A ou B) K	réponse (grammes) y
1	-	-	-	62
2	+	-	-	72
3	-	+	-	54
4	+	+	-	68
5	-	-	+	52
6	+	-	+	83
7	-	+	+	45
8	+	+	+	80

La représentation en ordre standard des variables codées tout en assurant une lisibilité facile des valeurs pour chaque expérience, nous permettra aussi de rendre plus rapide le calcul des effets des variables que nous présentons dans la section suivante.

2.3 Calculs des effets et interactions

Un design factoriel de 2^k expériences permet d'effectuer l'estimation de tous les k effets individuels (effets de premier-ordre), tous les $k(k - 1)/2$ effets provenant de l'interaction de deux facteurs, tous les $k(k - 1)(k - 2)/3!$ effets provenant de l'interaction de trois facteurs, etc... Si nous reprenons notre exemple précédent (table 2.2) et que nous désirons avoir l'effet de la température sur la réponse du système. Cela reviendrait à rechercher les changements au niveau de la réponse quand le facteur passe de son niveau bas ("-") à son niveau haut ("+") dans notre cas de 160 à 180 degrés.

Chaque estimation des effets est une valeur statistique de la forme:

$$\overline{y}_+ - \overline{y}_-$$

Celles-ci sont évaluées en effectuant une différence entre deux moyennes, chacune contenant 2^{k-1} observations, où y_+ représente la moyenne des réponses du système sous étude quand le facteur dont on recherche l'effet est à son niveau haut et y_- la moyenne des réponses quand le facteur est à son niveau bas. Pour l'exemple du réacteur nous aurons les effets de premier-ordre suivants:

Table 2.3.a Effet issu du changement de température (de 160° à 180 °C)

Calculs	Concentration	Catalyseur
y2 - y1 = 72-60 = 12	20	A
y4 - y3 = 68-54 = 14	40	A
y6 - y5 = 83-52 = 31	20	B
y8 - y7 = 80-45 = 35	40	B
Moyenne = 23		

La contribution moyenne de la température à la réponse du système est égale à 23.

Table 2.3.b Effet issu du changement de concentration (de 20 à 40%)

Calculs	Température	Catalyseur
y3 - y1 = 54-60= -6 y4 - y2 = 68-72= -4 y7 - y5 = 45-52= -7 y8 - y6 = 80-83= -3	160 180 160 180	A A B B
Moyenne = -5		

La contribution moyenne de la concentration à la réponse du système est de -5.

Table 2.3.c Effet isssu du changement de catalyseur (A ou B)

Calculs	Température	Concentration
y5 - y1 = 52-60= -8 y6 - y2 = 83-72= 11 y7 - y3 = 45-54= -9 y8 - y4 = 80-68= 12	160 180 160 180	20 20 40 40
Moyenne = 1,5		

La contribution moyenne du catalyseur à la réponse du système est de 1,5.

Avec cette méthode toutes les observations sont utilisées pour l'évaluation de chacun des effets et ces derniers sont définis avec la précision provenant de la différence entre quatre expériences.

L'effet moyen de la température est 23, mais en examinant les différentes expériences, nous pouvons remarquer que la température est plus grande avec le catalyseur B qu'avec le A. Les variables température et catalyseur interagissent entre elles parce qu'une fois unies elles ne se comportent pas de façon additive. La mesure de cette interaction est la différence entre la moyenne des effets de la température quand le catalyseur est au niveau A et la moyenne des effets de la température quand le catalyseur est au niveau B.

Pour l'interaction entre la température et le catalyseur nous aurons donc le calcul suivant:

Interaction T x K

catalyseur	moyenne des effets de la température
(+) B	33
(-) A	13
difference =	20

interaction T x K = $\frac{20}{2} = 10$ (où 2 représente le nombre d'observations)

interaction T x K = $\frac{1}{2}$(difference) = $\frac{33 - 13}{2} = 10$

$$= \frac{1}{4}(y_1 - y_2 + y_3 - y_4 - y_5 + y_6 - y_7 + y_8)$$

$$= \frac{y1 + y3 + y6 + y8}{4} - \frac{y2 + y4 + y5 + y7}{4}$$

ou

température (°C)	moyenne des effets du catalyseur
(+)180	11,5
(-)160	-8,5
difference	20

interaction T x K = $\frac{20}{2} = 10$

Les interactions T x C et C x K sont obtenus de la même façon.

concentration	moyenne des effets de la température	catalyst	moyenne des effets de la concentration
(+)40%	24,5	(+)B	-5,0
(-)20%	21,5	(-)A	-5,0
difference	3,0	difference	0,0

interaction T x C $= \frac{3}{2} = 1,5$ interaction C x K $= \frac{0}{2} = 0$

La dernière interaction à calculer sera l'interaction entre les trois facteurs de notre système. En considérant l'interaction (T x C), nous n'aurons qu'à calculer l'effet de celle-ci en fonction de la valeur du catalyseur. Puis calculer la moyenne de la difference entre les deux valeurs obtenues.

interaction T x C avec catalyseur B(+):

$$\frac{(y8 - y7) - (y6 - y5)}{2} = \frac{(80 - 45) - (83 - 52)}{2} = \frac{35 - 31}{2} = 2$$

interaction T x C avec catalyseur A(-):

$$\frac{(y4 - y3) - (y2 - y1)}{2} = \frac{(68 - 54) - (72 - 60)}{2} = \frac{14 - 12}{2} = 1$$

nous aurons donc

interaction T x C x K $= \frac{2 - 1}{2} = 0,5$

2.4 L'algorithme de Yates

L'Algorithme de Yates est une méthode permettant d'avoir l'estimation des effets de premier ordre et effets issus des interactions. Cette méthode s'applique à tous les designs factoriels à deux niveaux, mais aussi aux designs fractionnels factoriels que nous présenterons dans le chapitre suivant.

La construction d'une table d'analyse de Yates peut se décomposer en cinq étapes: (Hunter et al, 1988, 26.24)

étape 1: Organisation de la première colonne de la table en plaçant les valeurs d'initialisation en respectant l'ordre standard ou Yates order.

étape 2: Dans la deuxième colonne entrer les réponses du système correspondant à l'exécution de chacune des combinaisons figurant en colonne 1.

étape 3: Dans la première moitié de la colonne 3, entrer les valeurs correspondant à la somme des paires consécutives de la colonne 2. C'est-à-dire la première plus la seconde, la troisième plus la quatrième, ainsi de suite. Pour la seconde moitié de la colonne 3, on effectuera la différence entre les mêmes paires consécutives. Seconde valeur moins la première, quatrième moins la troisième, etc...

étape 4: Les colonnes 4, 5, ..., k + 2 seront obtenues en effectuant les opérations faites pour la colonne 3. C'est-à-dire pour chaque case obtenir la somme ou la différence des paires de la colonne précédente.

étape 5: Les valeurs de la dernière (colonne k + 2) représentent les estimations des effets de premier ordre et les effets des interactions avant la division par N/2. Seul la première valeur est divisé par N pour donner l'effet moyen général.

étape 6: La somme de toutes les réponses du système (colonne 2) doit être égale au total donné en première ligne de la colonne k+1.

étape 7: La somme des carrés de la colonne 2 doit être égale à la somme des carrés de la colonne k+2 divisé par 2^k.

Les étapes 6 et 7 sont des étapes de contrôle.

La table 2.4 nous montre la table de Yates correspondant à la table 2.2.a (exemple du réacteur) (Box, Hunter et Hunter, 1978, 322-324).

Table 2.4 Table de Yates de l'exemple du réacteur chimique donné dans la table 2.2.a

Expériences	T	C	K	réponses	(1)	(2)	(3)	estimation	facteurs
1	-	-	-	60	132	254	514	64,25	moyenne
2	+	-	-	72	122	260	92	23,0	t
3	-	+	-	54	135	26	-20	-5,0	c
4	+	+	-	68	125	66	6	1,5	tc
5	-	-	+	52	12	-10	6	1,5	k
6	+	-	+	83	14	-10	40	10,0	tk
7	-	+	+	45	31	2	0	0,0	ck
8	+	+	+	80	35	4	2	0,5	tck

Cette table de Yates peut s'interprèter de la manière suivante:

- en colonne 1 nous avons les numéros des expériences que nous avons conduites sur le modèle
- en colonne 2 la matrice de design représentant l'initialisation des facteurs pour chacune des expériences
- en colonne 3 nous avons l'indice de performance de chaque expériences
- les colonne 4, et 6 contiennent les résultats des calculs issus de l'étape 3 de conctruction d'une table de Yates (voir section 2.4)
- la colonne 7 contient les résultats déterminants les effets des facteurs et ceux de leurs interactions
- la dernière colonne contient les noms des facteurs intervenant dans chacun des effets

2.5 Tester les effets de premier ordre et les interactions

Une fois les effets obtenus il faut que l'expérimentateur prenne la précaution de tester l'importance statistique de ces derniers. Le test s'effectue en 5 étapes; (Hunter et al., 1988, 26.27-26.28)

étape 1: Choisir un niveau d'importance ß.

étape 2: S'il n'y a pas d'estimation de la variance due à des erreurs expérimentales, on peut procéder en utilisant la somme des carrés associées aux interactions entre trois ou plus de trois facteurs. La somme des carrés avec un degré de liberté pour un design factoriel de 2^k est égale à:

$$\frac{N(\text{effet})^2}{4}$$

où N représente le nombre d'observations, de réponses du système.

étape 3: Pour obtenir s^2, on divise le total de la somme des carrés obtenus à l'étape 2 par *e*, où *e* est le nombre d'intéraction contribuant au total. Dans un design factoriel de 2^k, le nombre d'interactions de troisième-ordre et plus est égal à: $2^k - (k^2 + k + 2)/2$.

étape 4: Regarder la valeur $t_{\beta/2}$ pour *e* degrés de liberté. Si une estimation de la variance est utilisé alors *e* est égale au degrés de liberté associé à l'estimation.

étape 5: L'erreur standard pour chaque estimation sera:

$$SE(\text{estimation de l'effet}) = SE(\bar{y}_+ - \bar{y}_-) = \frac{2s}{\sqrt{N}}$$

et l'intervalle de confiance 100(1-B) déterminera l'effet de la manière suivante:

$$\text{Effect} \pm t_{\beta/2}[\, SE(\text{effet})]$$

étape 6: Si la valeur absolue d'un effet est plus grande que SE(effet), on peut conclure que cet effet à une signification statistique.

Chapitre 3 - La méthode d'expérimentation fractionnelle factorielle

Etant donné k le nombre de variables de décisions et 2 le nombre de niveaux par variable, le nombre d'expériences requis pour effectuer un design factoriel total de 2^k augmente de façon géométrique quand k augmente. Les estimations des effets de premier ordre et les effets des interactions deviennent alors problématiques du fait du nombre d'expériences à conduire qui devient trop important. Nous montrerons dans les sections suivantes que l'on peut arriver à retrouver des estimations de ces effets en générant et évaluant uniquement un sous-ensemble des 2^k expériences nécessaires lors d'un design factoriel complet.

3.1 La redondance

Avec un design factoriel de k facteurs, le nombre de jeux d'essais constituant l'expérimentation est égal à 2^k expériences. Ces expériences nous permettraient d'estimer une moyenne de tous les effets, k effets de premier ordre, k(k-1) effets de second-ordre, k(k-1)(k-2) effet de troisième-ordre,..., un effet de k-facteur. Mais l'estimation de ces effets ne veut pas dire qu'ils sont tous de taille appréciable. Par exemple, les effets de premier ordre ont tendance à être plus large (amplitude) que les intéractions entre deux facteurs, qui à leur tour sont plus large que les intéractions entre trois facteurs, etc. Il y a tendance à avoir ce que l'on appelle redondance dans un design factoriel quand le nombre de facteur k devient important. Redondance causée par le nombre important des interactions estimées lui même dû, au nombre de facteur trop important (Box, Hunter et Hunter, 1978, 381). Le design fractionnel factoriel exploite cette redondance en cherchant à évaluer les estimations avec un nombre de facteur réduit. Nous illustrerons la méthode en donnant un exemple dans la section ci-après.

3.2 Le design fractionnel factoriel

Après avoir donné un énoncé du problème nous présenterons dans les sections suivantes, l'analyse factorielle complète du problème suivi de l'analyse du même problème mais cette fois-ci en utilisant un approche fractionnelle et pour terminer nous donnerons la méthode nous permettant de construire de façon systématique des designs fractionnels permettant de diviser par deux le nombre d'expériences requises lors d'un design factoriel complet.

3.2.1 Enoncé du problème

Pour cet exemple nous reprenons l'exemple du réacteur chimique augmenté cette fois-ci de deux variables quantitatives à savoir le taux de consommation et le taux d'agitation (Box, Hunter et Hunter, 1978, 377). Les tables 3.1.a et 3.1.b montrent le design factoriel de 2^5 expériences issu de cet exemple.

Table 3.1.a Design factoriel de 2^5 expériences
(valeurs d'initialisation)

Variables/Facteurs	Niveaux	
	-	+
1 Taux de consommation (litres/min)	10	15
2 catalyseur (%)	1	2
3 taux d'agitation (rpm)	100	120
4 temperature (C)	140	180
5 concentration (%)	3	6

Table 3.1.b Design factoriel de 2^5 expériences
Matrice de design

Exécution	Variables/Facteurs 1	2	3	4	5	réponses (% réaction) y
1	-	-	-	-	-	61
*2	+	-	-	-	-	53
*3	-	+	-	-	-	63
4	+	+	-	-	-	61
*5	-	-	+	-	-	53
6	+	-	+	-	-	56
7	-	+	+	-	-	54
*8	+	+	+	-	-	61
*9	-	-	-	+	-	69
10	+	-	-	+	-	61
11	-	+	-	+	-	94
*12	+	+	-	+	-	93
13	-	-	+	+	-	66
*14	+	-	+	+	-	60
*15	-	+	+	+	-	95
16	+	+	+	+	-	98
*17	-	-	-	-	+	56
18	+	-	-	-	+	63
19	-	+	-	-	+	70
*20	+	+	-	-	+	65
21	-	-	+	-	+	59
*22	+	-	+	-	+	55
*23	-	+	+	-	+	67
24	+	+	+	-	+	65
25	-	-	-	+	+	44
*26	+	-	-	+	+	45
*27	-	+	-	+	+	78
28	+	+	-	+	+	77
*29	-	-	+	+	+	49
30	+	-	+	+	+	42
31	-	+	+	+	+	81
*32	+	+	+	+	+	82

Les astérisques (*) représentent les choix arbitraires d'exécutions que l'expérimentateur désire effectuées.

3.2.2 Analyse factoriel complète

La table 3.1.b nous montre la matrice de design du design de 2^5 expériences en appliquant les formules que nous avons présentées dans le chapitre précédent ou en appliquant l'algorithme de Yates nous obtenons les résultats concernant les effets de chaque facteurs ou variables et ceux de leurs interactions sur la réponse du système. Ces effets sont répertoriés dans la table 3.2.

Table 3.2 Analyse du design factoriel de 2^5 expériences.
L'exemple du réacteur chimique (estimations des effets)

	Facteurs/Variables	Effets
Effets de premier-ordre	1	-1,375
	2	19,5
	3	-0,625
	4	10,75
	5	-6,25
Effets de second-ordre	12	1,375
	13	0,75
	14	0,875
	15	0,125
	23	0,875
	24	13,25
	25	2,0
	34	2,125
	35	0,875
	45	-11,0
Effets de troisième-ordre	123	1,50
	124	1,375
	125	-1,875
	134	-0,75
	135	-2,50
	145	0,625
	234	1,125
	235	0,125
	245	-0,250
	345	0,125
Effets de quatrième-ordre	1234	0,0
	1245	0,625
	2345	-0,625
	1235	1,5
	1345	1,0
Effet de cinquième-ordre	12345	-0,25

Cette table nous montre les effets des variables et ceux de leur interactions nous pouvons à partir de cette dernière, déterminer les plus importants en choisissant ceux ayant les valeurs absolues des effets les plus grandes: Donc les variables 2, 4, 5 et les interactions 24, et 45.

3.2.3 Analyse réduite: fractionnelle factorielle

Maintenant si l'on suppose qu'à cause de la taille importante de chacune des expériences (temps d'exécution), l'expérimentateur au lieu d'effectuer les 32 expériences désire n'en effectuer que 16. Les 16 expériences marquées d'un astérisque (*) dans la Table 3.1.b. Ces 16 expériences triées dans un certain ordre forme (ordre nous permettant de retrouver un design de 2^4 expériences) le design fractionel factoriel représenté dans la table 3.3.

Table 3.3 Analyse d'un design fractionnel factoriel de 2^{5-1} expériences

Exécution	1	2	3	4	5	12	13	14	15	23	24	25	34	35	45	réponses (%réaction)
17	-	-	-	-	+	+	+	+	-	+	+	-	+	-	-	56
2	+	-	-	-	-	-	-	-	-	+	+	+	+	+	+	53
3	-	+	-	-	-	-	+	+	+	-	-	-	+	+	+	63
20	+	+	-	-	+	+	-	-	+	-	-	+	+	-	-	65
5	-	-	+	-	-	+	-	+	+	-	+	+	-	-	+	53
22	+	-	+	-	+	-	+	-	+	-	+	-	-	+	-	55
23	-	+	+	-	+	-	-	+	-	+	-	+	-	+	-	67
8	+	+	+	-	-	+	+	-	-	+	-	-	-	-	+	61
9	-	-	-	+	-	+	+	-	+	+	-	+	-	+	-	69
26	+	-	-	+	+	-	-	+	+	+	-	-	-	-	+	45
27	-	+	-	+	+	-	+	-	-	-	+	+	-	-	+	78
12	+	+	-	+	-	+	-	+	-	-	+	-	-	+	-	93
29	-	-	+	+	+	+	-	-	-	-	-	-	+	+	+	49
14	+	-	+	+	-	-	+	+	-	-	-	+	+	-	-	60
15	-	+	+	+	-	-	-	-	+	+	+	-	+	-	-	95
32	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	82

A partir de la Table 3.3 l'expérimentateur peut calculer les effets des variables en utilisant soit le calcul directe soit la table d'analyse de Yates. Les effets provenant de cette table réduite sont présentés dans la Table 3.4.

Table 3.4 Analyse d'un design fractionel factoriel de 2^{5-1} expériences (estimation des effets)

	Facteurs/Variables	Effets
Effets de premier-ordre	1	-2,0
	2	20,5
	3	0,0
	4	12,25
	5	-6.25
Effets de second-ordre	12	1,5
	13	0,5
	14	-0,75
	15	1,125
	23	1,50
	24	10,75
	25	1,25
	34	0,25
	35	2,25
	45	-9,50

Une fois l'expérimentation réduite à 16 expériences effectuées, nous pouvons constater que les 15 effets de premier ordre et interactions entre deux facteurs obtenus ne sont pas très différent de ceux obtenus avec le design factoriel complet (Table 3.2). En fournissant la moitié des efforts, nous avons l'essentiel des informations à savoir que les facteurs 2, 4, 5 et les interactions 24 et 45 ont des répercussions plus importantes que les autres sur la réponse du système. Le lecteur peut constater qu'au niveau de la Table 3.4 nous avons introduit la notion de design de 2^{5-1} expériences pour désigner un design effectuant 16 expériences au lieu des 32 nécessaire dans un design de 2^5.

Nous montrons dans la section suivante qu'il existe un algorithme permettant d'obtenir de façon systématique ces fractions.

3.2.4 Construction d'un design fractionnel factoriel de 2^{k-1} expériences

Dans notre exemple nous avons considéré que l'expérimentateur avait choisit de façon arbitraire les 16 expériences qu'il voulait effectuer, il existe une méthode qui permet la définition de design fractionnel factoriel de 2^{k-1} expériences en deux étapes:
(Box, Hunter et Hunter, 1978, 381)

étape 1: Consiste en la génération de la matrice de design correspondant à un design factoriel de 2^{k-1} expériences où k représente le nombre de facteurs.

étape 2: Une fois la matrice générée et en ordre standard, la colonne représentant le facteur non considéré au cours de la création de la matrice, sera crée en effectuant le produit des signes de toutes les colonnes de la matrice. Le produit entre deux signes identiques donne un signe "+" tandis que des signes opposés donneront un signe "-".

Pour construire par exemple une design fractionnel factoriel de 2^{5-1} expériences nous devons:

1. Construire un design factoriel complet de 2^4 expériences avec les facteurs 1, 2, 3 et 4.
2. et définir la variable 5 égale au produit des colonnes des variables 1, 2, 3, et 4.

Ce design fractionnel factoriel de 2^{5-1} expériences a été obtenu de la manière suivante:

$$\frac{1}{2} 2^5 = 2^{-1}2^5 = 2^5 2^{-1} = 2^{5-1}$$

Cette notation permet de dire que nous avons un design composé de cinq variables chacune à deux niveaux, mais uniquement 2^{5-1} expériences seront utilisées pour déterminer les effets des facteurs de décisions. $2^{5-1} = 2^4 = 16$ expériences.

Avec ce design fractionnel factoriel de 2^{5-1} expériences, nous avons effectué 16 expériences et estimé 16 valeurs qui sont: la moyenne, les cinq effets de premier ordre et 10 effets issus d'interaction entre deux facteurs. Mais que sont devenus les 10 effets d'interaction entre trois facteurs les cinq effets entre quatre facteurs et l'effet entre cinq facteurs. C'est là qu'intervient le concept de redondance introduit dans la section précédente. Si nous voulons par exemple estimer l'effet issu de l'interaction entre trois facteurs, nous devons multiplier les signes des colonnes des trois facteurs. Pour l'interaction entre les facteurs 1, 2 et 3 on aurait:

123 or nous constatons que les signes de cette colonne sont égaux à ceux de la colonne 45

123	45
-	-
+	+
+	+
-	-
+	+
-	-
-	-
+	+
-	-
+	+
+	+
-	-
+	+
-	-
-	-
+	+

Donc 123 = 45, nous pouvons dire que les interactions 123 et 45 sont confondus.

3.3 Les Confondus

Avec la technique de design fractionnel factoriel il est très important de pouvoir obtenir les effets confondus sans utiliser la méthode de multiplication des signes qui, il faut le reconnaître, peut devenir très fastidieuse. Nous présentons dans cette section une méthode plus rapide (Box, Hunter et Hunter, 1978, 383).

Cette méthode utilise les quatre étapes suivantes:

étape 1: Les chiffres (e.g, 3, 12) font références aux colonnes de signes plus et moins (une colonne pour chaque facteur).

étape 2: Un produit de colonne est égal à la multiplication des éléments constituant cette colonne.

étape 3: En multipliant les éléments de n'importe quel colonne avec des éléments d'une colonne identique nous obtenons une colonne de signes plus, que nous désignerons comme étant égale à I (comme identité), c'est-à-dire $1 \times 1 = 1^2 = I$.

étape 4: Un confondu comme C_{45} est obtenu en multipliant les observations (réponses du système) par les signes plus et moins de la colonne 45 et en les divisant par N/2 où N représente le nombre d'observations ou d'expériences.

Dans notre exemple du réacteur le design de 2^{5-1} expériences a été obtenu en définissant la colonne 5 comme étant égale à 1234 cette relation est le générateur du design. En multipliant les deux cotés par 5 nous obtenons:

$$5 \times 5 = 1234 \times 5$$

ou $$5^2 = 12345$$

donc le générateur du design est équivalent à $I = 12345$

Notre design a été défini avec un seul générateur, donc la relation est $I = 12345$.

Cette relation permet de définir les confondus de notre expérimentation.

$$I = 12345$$
$$I \times I = 12345 \times I$$
$$I = I^2 2345$$
$$I = 2345$$

Nous avons pu ainsi obtenir les estimations et confondus de la table 3.5.

Table 3.5 Confondus et estimations du design fractionnel factoriel de 2^{5-1} expériences de la table 3.3

relation entre paire de colonne	confondus	estimations
1=2345	1+2345	-2,0
2=1345	2+1345	20,0
3=1245	3+1245	0,0
4=1235	4+1235	12,25
5=1234	5+1234	-6,25
12= 345	12+ 345	1,5
13= 245	13+ 245	0,5
14= 235	14+ 235	-0,75
15= 234	15+ 234	1,25
23= 145	23+ 145	1,5
24= 135	24+ 135	10,75
25= 134	25+ 134	1,25
34= 125	34+ 125	0,25
35= 124	35+ 124	2,25
45= 123	45+ 123	-9,5
I = 12345 = 65,25		

Ces confondus veulent dire que même si nous n'avons pas les effets de troisième-ordre et les effets de quatrième ordre, nous pouvons retrouver la valeur de leur effets en effectuant les opérations suivantes:

Si nous recherchons par exemple l'effet de l'interaction 2345, nous n'aurions qu'à résoudre l'équation suivante:

$$x + y = -2,0$$

où x représente l'effet de 1 et y l'effet de 2345

$$-1,375 + y = -2,0$$

$$y = -2,0 + 1,375$$

$$y = -0,625$$

pour contrôle nous pouvons nous référer à la table 3.2 qui montre que l'effet de l'interaction 2345 est bien égal à -0,625. Pour de plus amples renseignements le lecteur pourra se référer à Box, Hunter et Hunter (1978) et Hunter et al. (1988).

Chapitre 4 - Les réseaux de Pétri et le langage Voltaire

Nous présentons dans ce chapitre les réseaux de Pétri et réseaux de performances pour ensuite présenter le langage Voltaire qui est langage de simulation permettant la description de ces réseaux. Pourquoi introduisons nous la notion de réseaux de Pétri ? Comme on le verra dans le chapitre 5, nous présenterons le logiciel que nous avons développé, celui-ci bien que pouvant virtuellement traiter des modèles de simulation décrit dans des langages de simulation quelconques et effectuer tout type d'analyse, a été implanté pour traiter des modèles décrit à l'aide du langage Voltaire. Aussi pour aider quelques peu le lecteur dans la comprehension de ce langage, nous avons jugé bon de présenter ces réseaux de Pétri.

4.1 Les réseaux de Pétri

Définition 4.1: Un réseau de Pétri est un triplet RP = (P, T, A) où

$P = \{ p_1, p_2, \ldots, p_m \}$ est un ensemble de places. $(P \neq \emptyset)$;

$T = \{ t_1, t_2, \ldots, t_n \}$ est un ensemble de transitions. $(T \neq \emptyset)$;

$A \subseteq \{P \times T\} \cup \{T \times P\}$ est un ensemble d'arcs orientés.

Définition 4.2: Le marquage M d'un réseau de Pétri RP = (P, T, A) est une fonction

$$M : P \rightarrow N.$$

où N est l'ensemble des entiers naturels.

Le marquage d'un réseau de Pétri est défini à l'aide du marquage de ses places. La notation $M(p_i)$ est utilisée pour définir le marquage d'une place p_i. Le marquage d'une place est défini au niveau graphique par les jetons se trouvant à l'intérieur des places (Figure 4.1). $M(p_i) = 4$ veut dire que nous avons quatre jetons à la place p_i.

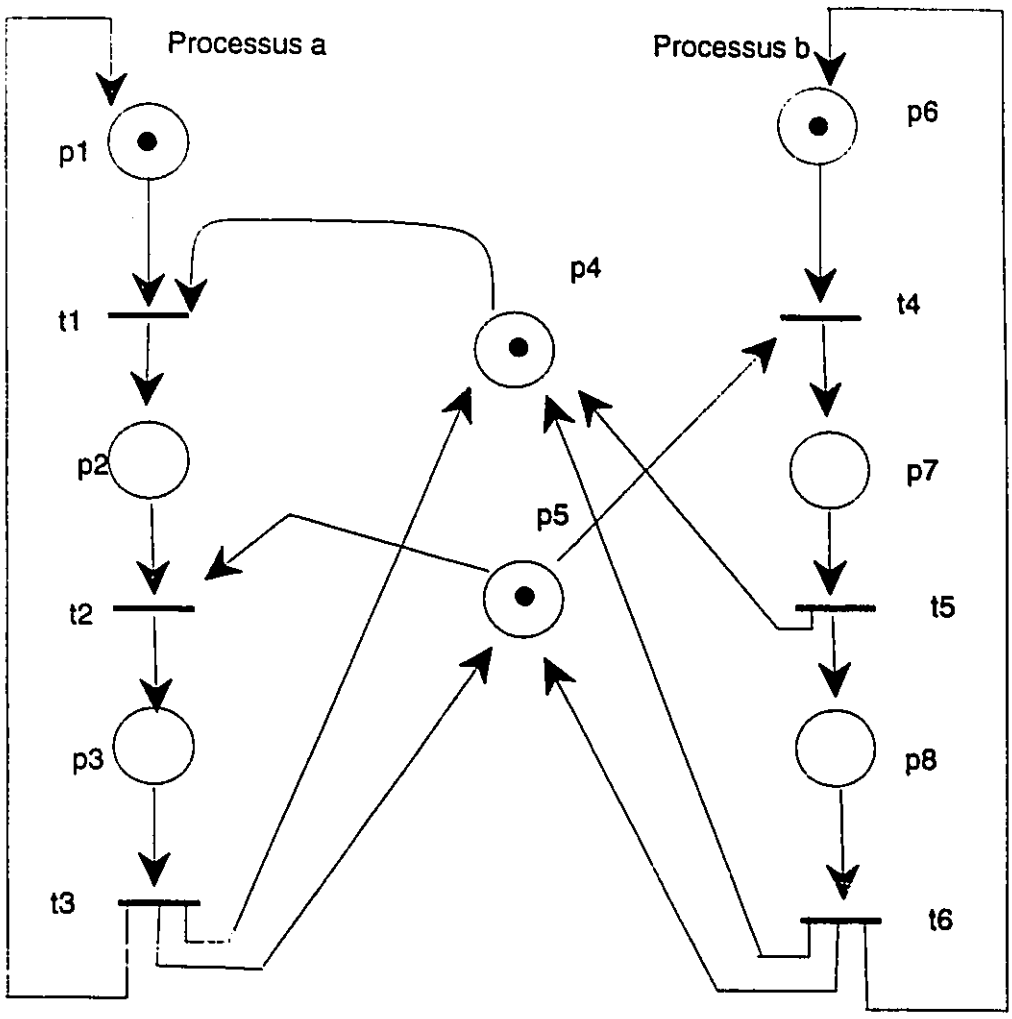

Figure 4.1 Réseau de Pétri représentant deux processus concurrents

Définition 4.3: L'état initial d'un réseau de Pétri RP = (P, T, A), c'est-à-dire le nombre de jetons présent dans les différentes places est appelé le marquage initial.

Le marquage d'un réseau de Pétri et son marquage initial sont des notions très importantes dès le moment où l'on veut étudier la nature dynamique du système modélisé.

En effet le marquage représente l'état du système à un moment particulier, ce qui veut dire que l'état d'un réseau de Pétri change quand celui-ci passe d'un marquage à un autre. Ces changements d'états sont définis par l'exécution du système modélisé. Un réseau de Pétri permet donc de modéliser la structure statique d'un système mais aussi son comportement dynamique. Le passage d'un état (marquage) à un autre obéit à certaines règles qui sont décrites dans les définitions ci-après.

Définition 4.4: Soit un réseau de Pétri RP = (P, T, A)

1. Pour chaque transition $t_j \in T$,

   $I(t_j)$ est l'ensemble des places d'entrée et

   $O(t_j)$ est l'ensemble des places de sortie défini par:

   $$I(t_j) = \{ p_i \mid (p_i, t_j) \in A \},$$

   $$O(t_j) = \{ p_i \mid (t_j, p_i) \in A \}.$$

2. Pour chaque $p_i \in P$,

   $I(p_i)$ est l'ensemble des transitions d'entrée et

   $O(p_i)$ est l'ensemble des transitions de sortie défini par:

   $$I(p_i) = \{ t_j \mid (t_j, p_i) \in A \},$$

   $$O(p_i) = \{ t_j \mid (p_i, t_j) \in A \}.$$

Définition 4.5: Une transition $t_j \in T$ d'un réseau de Pétri RP = (P, T, A) est exécutable si et seulement si $M(p_i) > 0, \forall p_i \in I(t_j)$.

Définition 4.6: Une transition exécutable tj ∈ T d'un réseau de Pétri RP = (P, T, A) peut *être déclenchée* si et seulement si elle est exécutable. Le *tir* provoque un changement de l'état du réseau de Pétri défini par:

$$M'(p_i) = \begin{cases} M(p_i) + 1 \text{ si } p_i \text{ n'appartient pas à } I(t_j) \text{ et } p_i \in O(t_j) \\ M(p_i) - 1 \text{ si } p_i \in I(t_j) \text{ et } p_i \text{ n'appartient pas à } O(t_j) \\ M(p_i) \text{ sinon} \end{cases}$$

Définition 4.7: Soit le réseau de Pétri RP = (P, T, A) celui-ci peut être considéré comme un graphe bi-partie dans lequel les places sont représentées par des cercles, les transitions par des barres verticales et les jetons par des points noirs à l'intérieur des places. Ces éléments graphiques sont reliés entre eux par les arcs constituant l'ensemble A (Figure 4.1).

Les réseaux de Pétri du fait de la possibilité qu'ils ont de représenter la structure statique et dynamique d'un système sont des outils de modélisation très puissants. L'analyse d'un système modéliser à l'aide d'un réseau de Pétri revient en fait à l'analyse du réseau de Pétri lui-même. Cette analyse utilise le concept d'ensemble d'états atteints. L'ensemble des états atteints d'un réseau de Pétri est l'ensemble de tous les marquages que l'on peut atteindre à partir du marquage initial, qu'il soit fini ou infini.

Les réseaux de Pétri ont certaines restrictions quand au niveau de l'analyse de la performance d'un système. Ces limitations sont en générales les suivantes:

1. Il n'y a aucune notion de temps. Dans un réseau de Pétri, les *déclenchements* des transitions sont instantanés.

2. Limitation au niveau du fait que pour qu'une transition soit exécutable il faut que toutes les places d'entrées aient un marquage supérieur à zéro.

3. Les jetons ne peuvent pas représenter des données réelles car ils ne peuvent pas transporter d'informations avec eux.

4. Il n'existe qu'un seul type d'arcs n'obéissant qu'à une seule règle de *déclenchement* quand une transition est exécutable. Ces arcs n'ont pas de notions de poids par exemple.

Les réseaux de Pétri de performances sont des dérivés des réseaux de Pétri qui ont été définis pour pallier à ces limitations. Ces réseaux de Pétri de performances peuvent être divisés en deux groupes si l'on se base sur la notion de temps qu'ils utilisent. Le premier groupe est composé de réseaux utilisant des temps d'exécution constants au niveau des transitions: ce sont les réseaux de Pétri déterministique. Le second groupe est composé par les réseaux utilisant des temps stochastiques au niveau de l'exécution des transitions: ce sont les réseaux de Pétri stochastique. Le logiciel Voltaire que nous présentons dans la section suivante peut-être classé dans les deux groupes puisqu'il peut utiliser soit des temps déterministiques, soit des temps stochastiques au niveau des transitions.

4.2 Le logiciel Voltaire

Voltaire est un logiciel de simulation basé sur les réseaux de Pétri ayant la notion de temps dans leur structure (Parent, 1990). Ce logiciel est composé de deux parties: un simulateur et un langage de description hiérarchique de réseaux de Pétri. Le simulateur qui occupe très peu de place a été défini en vue de simuler efficacement les modèles les plus compliqués. Le langage de description bien que simple permet de représenter des systèmes complexes en utilisant un nombre modéré d'instructions.

Les objets dans Voltaire sont des éléments provenant d'un réseau de Pétri de performance. Comme dans un réseau de Pétri nous avons des places, des transitions et des jetons, mais aussi des mémoires globales et des sémaphores permettant la modélisation de processus de synchronisation et d'allocation de ressources. Les jetons peuvent avoir des attributs définis par l'utilisateur qui peuvent être controllés par les transitions.

Les variables présentes dans la mémoire globale permettent de changer la fonctionnalité d'un réseau uniquement en changeant le contenu d'une variable. La description de modèles complexes est simplifiée par le fait que le programmeur du modèle puisse les définir de façon hiérarchique en niveaux différents. Au niveau le plus haut de la hiérarchie nous avons la définition de blocs pouvant contenir d'autres blocs connectés les uns aux autres. Au niveau le plus bas nous avons des blocs contenant les primitives du langage c'est-à-dire les places, transitions, sémaphores et mémoires globales. Ce langage de description reste ouvert pour permettre la modélisation en C++ de certains détails difficilement modélisable avec des réseaux de Pétri.

Les extensions du langage par rapport aux réseaux de Pétri conventionnels sont les suivantes:

- Transitions plus complexes (conditions au niveau des places et variables mémoires).
- Les jetons peuvent avoir un nombre arbitraire d'attributs.
- Variables mémoires.
- Sémaphores avec priorités.
- Places avec des transitions de sortie multiple qui ont 3 disciplines de sélection aléatoire uniforme, aléatoire par poids et par priorité.
- Hiérarchie: les blocs sont composés de blocs.
- Flexibilité et ouverture: Les transitions peuvent être programmées en C++.
- Modularité.
- Ré-utilisation: Les blocs peuvent être mis dans des librairies partageables.

4.2.1 Les domaines d'applications de Voltaire

Voltaire a été défini en vue de permettre la modélisation et la simulation de systèmes complexes et/ou réguliers pouvant être modélisés à base d'évènements discrets. Les domaines que nous pouvons citer sont les suivants:

- Evaluation de la performance de réseaux d'ordinateurs.
- Evaluation et design de protocoles et systèmes de communication.
- Prototypage de logiciels système.
- Performance de systèmes matériel et logiciel.

4.2.2 Un réseau de Pétri sous Voltaire

Nous définissons un réseau de Pétri constitué d'une place et d'une transition. Le marquage initial est représenté par le jeton se trouvant dans la place P1. L'arc partant de la transition t1 et n'ayant aucune connection, signifie la fin du traitement sous Voltaire. Figure 4.2 (Des exemples plus complexes seront présentés dans le chapitre 6)

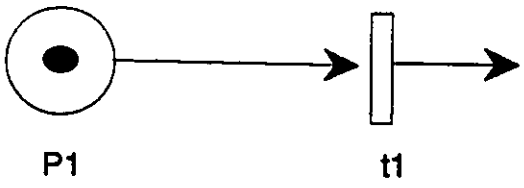

Figure 4.2 Un réseau de Pétri sous Voltaire

L'exécution de ce réseau est la suivante : la transition t1 tire quand la condition de sa place d'entrée est vérifiée, elle consomme alors le jeton se trouvant dans la place p1. Après un certain temps défini par la transition, t1 détruit le jeton et devient libre. L'exécution s'arrete car il n'y a plus de jetons dans la place p1.

La description de ce réseau de Pétri dans le langage Voltaire est présentée dans la Figure 4.3.

```
SIMULATION_PARAMETERS:
          BEGIN_TIME          =          0;
          END_TIME  =         100;
          TRACE               =          TRACE_ALL;

// définition des places

PLACES:
          <p1>

// définition des transitions

TRANSITIONS:

//  t1 est composé de quatre phases

          <t1>
                    PLACE_CONDITION:TOKEN_IN:x(p1);
                    CONSUMPTION:          GET(x);
                    DELAY:              SET_DELAY(5);
                    PRODUCTION:           SINK(x);

// marquage initial: 1 jeton dans p1

INITIAL_MARKING:

          TYPE=token, PLACE=p1;
```

Figure 4.3 Exemple de programme Voltaire

Pour de plus amples renseignement concernant le logiciel Voltaire le lecteur pourra se réferer à Parent (1990).

Chapitre 5 - Le logiciel SimAd-Voltaire

L'objectif de cette thèse était d'une part la conception de l'architecture d'un environnement de simulation permettant le design, la conduite d'expérimentation et l'analyse de modèle de simulation par des méthodes statistiques. Et d'autre part l'implantation de cette architecture. Cette architecture nommée par SimAd (Simulation Advisor) est applicable pour différents languages de simulation et méthodes de design. Le logiciel nommé par le nom SimAd-Voltaire est implementé pour traiter des modèles décrits avec le language Voltaire (Parent, 1990) et analyser lesdits modèles en utilisant les méthodes factorielles et fractionnelles factorielles. Nous présentons dans ce chapitre le logiciel que nous avons développé en décrivant sa conception, son implantation, son utilisation.

5.1 La conception du logiciel SimAd-Voltaire

Nous rappelons que le but de la simulation est l'étude du comportement du modèle dans des conditions d'expériences différentes. Cette comparaison se fait en fonction de la valeur prise par la réponse du modèle. Cette réponse appelée aussi indice de performance doit être spécifié de manière explicite par l'utilisateur du logiciel SimAd-Voltaire (section 5.2.5). Sans cet indice de performance aucune analyse des effets des paramètres ne serait possible. Nous avons définit au niveau de la conception de ce logiciel six modules ayant des tâches distinctes mais complémentaires. Ces modules décrits dans les sections suivantes, sont présentés dans la figure 5.1.

5.1.1 Interface

Ce module d'interface entre l'utilisateur et le programme devra faciliter l'utilisation du système par l'analyste. Certaines personnes ont tendance à négliger ce module, mais son importance est relativement grande car de lui dépendra les communications entre l'analyste et notre système.

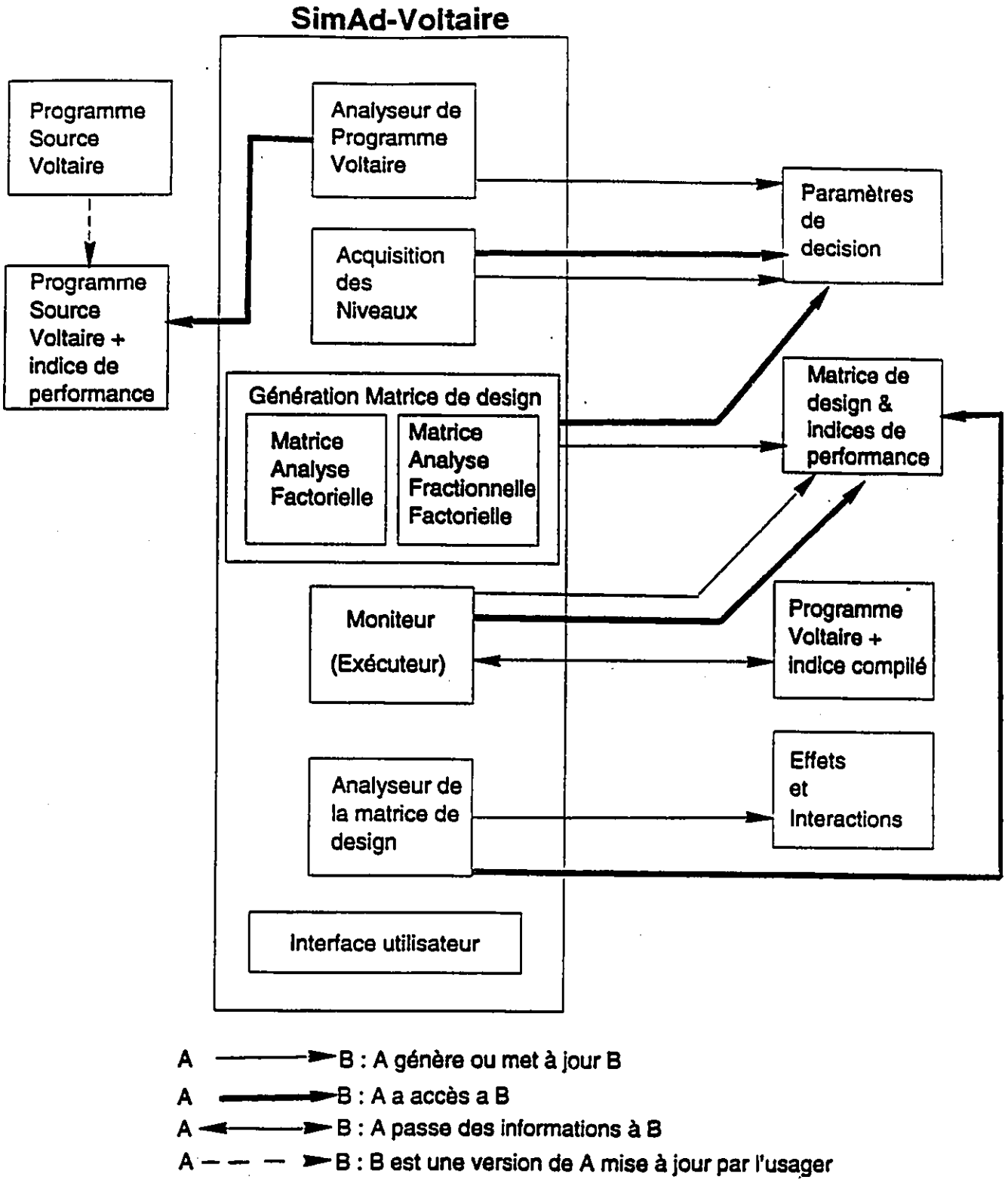

Figure 5.1 Architecture du logiciel SimAd

5.1.2 Analyseur de programme

Ce module devra chercher dans le programme source les paramètres de décisions que l'analyste désire étudier. Ce module permet de rendre SimAd indépendant d'un langage de simulation. En effet, pour traiter des modules décrits en Simscript par exemple, il suffira d'intégrer au logiciel un analyseur de programme Simscript), de même des analyseurs spéciaux pour des modèles décrits en SLAM II, GPSS, etc.

5.1.3 Module d'acquisition des niveaux des variables

Ce module permettra d'attribuer de façon intéractive les valeurs critiques pour chaque paramètres de décisions. Dans le cas de SimAd, nous aurons deux niveaux par paramètres. Mais en fonction du type d'analyse effectuée sur le modèle, ce module peut en théorie permettre l'acquisition de 1 à N (entier) niveaux.

5.1.4 Générateur de la matrice de design

Ce module crée la matrice de design qui contient les valeurs qu'auront les paramètres avant chaque exécution. Ce générateur dans notre cas utilisera deux types de méthodes soit une génération factorielle, soit une fractionnelle factorielle. De ce module dépend la méthode d'expérimentation. En conséquence pour permettre la réalisation de différentes méthodes, il suffira de les rajouter dans ce module.

5.1.5 Moniteur ou exécuteur

Ce module exécutera la version compilée du programme source avec en entrée une ligne de la matrice de design dans le cas de SimAd. Nous rappelons qu'une ligne de la matrice de design correspond à une expérience. Au niveau de ce module, nous devrons effectuer les opérations suivantes:

- communication automatique des valeurs des paramètres d'entrées au programme compilé
- exécution du programme

- récupération de l'indice de performance ou réponse
- rangement de cet indice dans la matrice de design.

5.1.6 Analyseur de la matrice de design

Ce module tout comme le module de génération de la matrice suit les conventions définies par les méthodes factorielles et fractionnelles factorielles. Il permettra le calcul des effets de premier-ordre et celui des effets des interactions entre facteurs, l'estimation de la variance, et effectuera aussi des tests d'importance au niveau des effets obtenus.

5.2 L'implantation du logiciel SimAd-Voltaire

L'implantation du logiciel s'est faite en respectant l'architecture que nous avons présentée dans la figure 5.1. SimAd a été développé en langage C sur une station de travail SPARC et sous le système d'exploitation UNIX. Ce logiciel de 6000 lignes de codes a été implanté en respectant une structuration rendant les modifications et sa maintenance aisées.

5.2.1 Implantation de l'interface

L'interface de SimAd a été implanté en utilisant les outils offerts par X-Windows et Xview. Nous avons définis des boutons, des menus déroulants, des fenêtres de type texte et de type numérique. Cette interface graphique permet une utilisation facile du logiciel, parce que l'activation des opérations se fait à partir de la souris. La figure 5.2 nous présente le premier écran obtenu après avoir lancé l'exécution du logiciel.

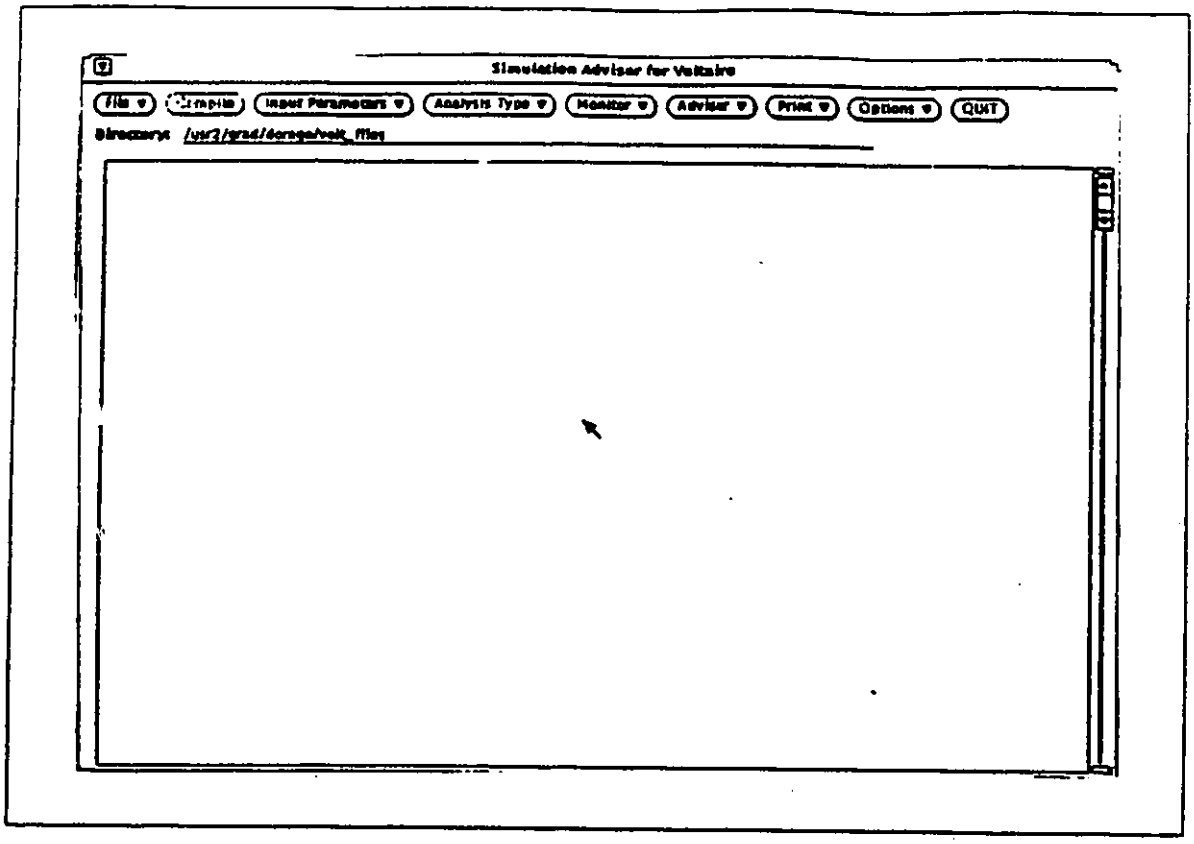

Figure 5.2 Ecran principal de SimAd-Voltaire

5.2.2 Implantation de l'analyseur de SimAd-Voltaire

A ce niveau, nous parlons d'analyseur de SimAd-Voltaire parce qu'il analyse des modèles de simulation uniquement décrits à l'aide du langage Voltaire. Le langage Voltaire a la particularité de permettre la définition de paramètres de décision prenant leur valeur d'initialisation de l'environnement d'exécution ou d'un fichier externe de type texte. Cette particularité permet l'exécution d'un modèle dans différentes conditions sans passer par une phase de recompilation ou de lecture des paramètres par l'utilisateur. Un paramètre recevant une valeur de l'environnement (Shell sous Unix) est défini sous Voltaire à l'aide du caractère "@" préfixant une variable.

Donc au niveau de la construction du programme Voltaire, le programmeur devra prendre la précaution de définir tous ses paramétres de décision de la manière suivante:

- nombre_de_serveur deviendra sous Voltaire
- @nombre_de_serveur.

L'analyseur de programme Voltaire recherche toutes les variables commençant par "@". Une fois ces variables trouvées, il les insère dans une liste chainée défini à cet effet. Cette liste, vu le nombre de facteurs, sera une liste à chainage avant uniquement. Le code correspondant à l'analyseur est décrit dans la figure 5.3. L'implantation de SimAd-Voltaire permet de gérer au maximum 26 paramètres de décisions. Ce nombre est largement suffisant puisqu'avec un modèle ayant 26 paramètres sous études l'analyste devrait exécuter 2^{26} expériences avant d'avoir les estimations des effets dans un design factoriel.

```
void
analyseur_voltaire()
{
/* recherche des paramètres de décision */
 Openfile();
 /*
       fonction permettant l'ouverture du fichier
       contenant le programme Voltaire
 */
 while (!feof(file_in_vol))
 {
     ch = Search_Next_char(file_in_vol, '@');
          /* fonction recherchant la prochaine occurence
             du caractère '@'
          */

     if (ch == '@')
     {
        i=0;
        strcpy(var_inter, BLANK); /* remise à blanc de la
                                     variable BLANK
                                  */
        do
```

```
        {
           fscanf(file_in_vol, "%c", &ch);
           var_inter[i] = ch;
           i++;
        } while (ch != special_char());
        var_inter[i-1] = ' ';
        /* var_inter contient le nom du paramètre de décision */

        /* Ajout de la variable dans la liste des variables */
        first_var = Add_New_Var (first_var, var_inter);

    }
 wrk_var = first_var; /* Pointe sur la première variable */
 factor  = Count_List_Var(first_var); /* dètermine le nombre de
                                         facteurs
                                      */
 fclose (file_in_vol); /* ferméture du fichier programme */

}
```

Figure 5.3 Codes de l'analyseur de programme Voltaire

5.2.3 Implantation du module d'acquisition des niveaux

Ce module utilise la liste des variables citées dans la section précédente. La structure de cette liste est donnée dans cette section parce que tout les champs constituant cet enregistrement seront gérer par le module d'acquisition des niveaux. Puisque nous implantons les méthodes factorielles et fractionnelles factorielles à deux niveaux, nous aurons besoin de deux champs, l'un désignant le niveau d'initialisation bas et l'autre le haut. La structure de cette liste est présentée dans la Figure 5.4.

```
struct variables_list
{
 char   var_name[MAX_LEN_VAR]; /* nom de la variable */
 char   lower[MAX_LEN_VAR]; /* valeur niveau bas */
 char   upper[MAX_LEN_VAR]; /* valeur niveau haut */
 int    set_flag; /* initialisation (oui/non) */
 struct variables_list *next_var; /* Pointe sur la variable
                                     suivante
                                  */
} list_var,
  *first_var,  /* Pointe sur la première variable */
  *last_var;   /* Pointe sur la dernière variable */
```

Figure 5.4 Structure de la liste des variables

L'implantation de ce module consiste en la réalisation de plusieurs étapes:

- Définition d'une fenêtre présentant à l'utilisateur la liste des paramètres de décision du modèle sous étude
- Sélection par l'utilisateur d'un paramètre dans la liste
- Ouverture d'une fenêtre présentant les valeurs d'initialisation du paramètre (si elles existent)
- Mise à jour de la liste avec les nouvelles valeurs données par l'utilisateur.

La Figure 5.5 nous décrira uniquement la phase de mise à jour des valeurs d'initialisation.

```
int
set_range_done(item, event)
Panel_item   item;
Event        *event;

/* acquisition des niveaux des variables */

{

 char                 lower_val[MAX_LEN_VAR],
                      upper_val[MAX_LEN_VAR];
 struct variables_list *wrk_var;
 strcpy(lower_val, (char *) panel_get_value(panel_lower));
```

```
strcpy(upper_val, (char *) panel_get_value(panel_upper));

/* recherche de la variable dans la liste */

wrk_var = first_var;  /* Pointe sur la première variable */

while ((strcmp(wrk_var->var_name, fact_sel) != 0) &&
       (wrk_var != NULL))
{
      wrk_var = wrk_var->next_var;
}
if (strcmp(wrk_var->var_name, fact_sel) == 0)
{
   /* la variable a été trouvée */
   strcpy(wrk_var->lower, lower_val); /* mise à jour */
   strcpy(wrk_var->upper, upper_val); /* mise à jour */
   wrk_var->set_flag = 1;             /* initialisation oui */
}
xv_set(panel_range, PANEL_INACTIVE, FALSE, NULL);
xv_set(panel_list , PANEL_INACTIVE, FALSE, NULL);

xv_destroy_safe(frame_text);
return XV_OK;
}
```

Figure 5.5 Mise à jour des variables

Nous précisons que les variables "upper" et "lower" sont de type texte uniquement pour faciliter les contrôles au niveau des types.

5.2.4 Implantation du générateur de la matrice de design

Ce module au niveau de SimAd-Voltaire générera l'ensemble des expériences qu'il y aura à conduire sur le modèle sous étude. Le nombre de paramètres des modèles de simulation n'étant pas constant, nous utilisons une structure nous permettant de définir une matrice de taille dynamique. La structure de la matrice est présentée dans la Figure 5.6.

```
typedef struct matrix
{
   char           mat_elt;
   unsigned short row_nbre;
   unsigned short col_nbre;
   struct matrix *prev_row, *next_row, *prev_col, *next_col;
} Matrix;
```

Figure 5.6 Structure de la matrice de design

Cette structure nous permet de générer des matrices de taille dynamique. Dans le cas de SimAd-Voltaire la matrice contenant les codes d'initialisation des variables (matrice de design) aura les dimensions suivantes:

- le nombre de lignes = $2^{(\text{nombre de variables})}$
- le nombre de colonnes = nombre de variables.

Ces dimensions sont les mêmes dans le cas d'une analyse fractionnelle, c'est le nombre de lignes qui diminuera en fonction de la fractionnalisation. Le seul problème au niveau de la méthode fractionnelle est la détermination des produits des colonnes à effectuer pour avoir les codes des colonnes manquantes. La Figure 5.7 illustre l'exemple d'un design fractionel factoriel de 2^{5-2} expériences, c'est-à-dire l'évaluation d'un modèle de 5 variables par 2^3 expériences au lieu des 32 à faire lors d'un design factoriel complet. Avec la méthode fractionnelle, la génération des codes d'initialisation des colonnes des variables 4 et 5 (les variables manquantes) se fera en effectuant le produit des colonnes 1 et 2 pour la colonne de la variable 4 et le produit des colonnes 2 et 3 pour la colonne 5.

Variable	1	2	3	4=12	5=23
	-	-	-	+	+
	+	-	-	-	+
	-	+	-	-	-
	+	+	-	+	-
	-	-	+	+	-
	+	-	+	-	-
	-	+	+	-	+
	+	+	+	+	+

Figure 5.7 Design fractionel factoriel 2^{5-2}

L'implantation de la méthode consistant à effectuer les produits de certaines colonnes pour en obtenir d'autres a été faite en utilisant de fichier de type texte. Pour un design de 2^{5-2} expériences le programme s'exécutera de la manière suivante:

- Génération d'une matrice de 8 lignes et 5 colonnes.
- Génération des codes en ordre standard pour les trois premières colonnes. ($2^{5-2}=2^3$)
- Pour les colonnes supplémentaires (4 et 5), le programme recherchera dans le répertoire de travail un fichier texte ayant pour nom FRACT05_02.DAT. Ce fichier contient les informations permettant d'effectuer le produit des colonnes, ces informations sont le numéro de la colonne à générer suivi du nombre de colonnes dont il faudra faire le produit et enfin les numéros des différentes colonnes intervenant dans ce produit.

Nous avons choisit cette structure pour permettre à l'analyste de changer à tout moments, ses critères de génération de colonnes manquantes. La sélection du type d'analyse est laissée à l'utilisateur.

Au niveau de la méthode fractionnelle factorielle nous nous sommes limité à l'implantation de fractions générant:

- 4 exécutions
- 8 exécutions
- 16 exécutions
- 32 exécutions
- 64 exécutions
- 128 exécutions
- 256 exécutions

et 512 exécutions

5.2.5 Implantation du moniteur ou exécuteur

Le moniteur permet d'effectuer soit une exécution sélective des combinaisons, soit une exécution de l'ensemble des combinaisons présentes dans la matrice de design. Quelle soit automatique ou sélective cette exécution suivra les étapes suivantes:

- Récupération de la ligne de la matrice de design sélectionnée
- Pour chacun des paramètres de décisions:
  * récupérer la valeur d'initialisation dans la liste des variables.
    (Valeur de niveau bas si le code est "-", de niveau haut sinon)
  * écrire dans un fichier texte ayant comme nom, le nom du fichier source Voltaire avec comme extension VAR, la commande:
    SETENV nom du paramètre valeur d'initialisation
    (setenv est une commande UNIX permettant la définition d'une variable recevant sa valeur de l'environnement).
- Le fichier texte créé précedemment sera lu par le programme Voltaire compilé en tapant la commande:
  nom_programme_compilé nom_fichier.VAR

- Aprés l'exécution du programme, l'indice de performance sera mis dans la première colonne d'une matrice ayant les mêmes dimensions que la matrice contenant les codes d'initialisation. La structure de la nouvelle matrice est la même que la précédente mise à part le type du champ "mat_elt" qui devient "double" pour pouvoir stocker les réponses du système. La récupération de l'indice de performance se fait grâce a un fichier texte que le développeur du programme Voltaire doit absolument créer et y écrire la valeur de la réponse du système à la fin de l'exécution de son programme. Ce fichier doit avoir pour nom PERFORM.DAT. La création d'un tel fichier se fait en utilisant la syntaxe du langage C ou C++. Le lecteur pourra se référer à la partie concernant l'utilisation de SimAd-Voltaire.

5.2.6 Implantation de l'analyseur de la matrice de design

L'analyseur est le module permettant le calcul des effets des paramètres. Le calcul des effets des paramètres est effectués grâce à l'implantation de l'algorithme de Yates. Cette implantation utilise la matrice dans laquelle nous avons stocké les différentes réponses du modèle. La figure 5.8 nous montre le code tel qu'il à été implanté. Cette implantation suit exactement la méthode dans la section 2.4

```
void
menu_yates_table()
/* Algorithme de yates */
{
   struct variables_list *wrk_var;
   int                   cpt, i;
   char                  *buf;
   int                   test_ret, k, j, rep_div;
   int                   perf_ok=0, result;
   Matrix                *ptr_wrk;
   Matrix_run            *ptr_perf_ind, *ptr_wrk_run, *ptr_1,
                         *ptr_2, *ptr_3,
                         *ptr_wrk_start_run;
   Frame                  frame_yates;

  k = factor;     /* Nombre de facteur */
  k++;            /* 2 colonnes supplémentaires */

  for (j=2; j<=k; j++)    /* colonne 2 jusqu'à k  */
  {
    i=1;
    cpt=1;
    ptr_wrk_run = first_perf;  /* pointe l'indice de performance
                                  de la première combinaison */

   while ( i<= (max_line/2) )  /* première moitié  Etape 3 */
   {
     ptr_1 = Goto_elt_double(i,j, max_line, factor+2, first_perf);
    ptr_2  = Goto_elt_double(cpt,j-1, max_line, factor+2,first_perf);
    ptr_3  = Goto_elt_double(cpt+1,j-1, max_line, factor+2,first_perf);
    ptr_1->mat_elt    = ptr_2->mat_elt + ptr_3->mat_elt;

    cpt = cpt+2;
    i++;
   }
   rep_div = (int)(max_line / 2);
   i = rep_div+1;
   cpt =1;
   while (i<=max_line)        /* seconde moitié  Etape 3 */
   {
    ptr_1 = Goto_elt_double(i,j, max_line, factor+2,first_perf);
    ptr_2 = Goto_elt_double(cpt+1,j-1 ,max_line, factor+2,first_perf);
    ptr_3 = Goto_elt_double(cpt,j-1, max_line, factor+2,first_perf);
    ptr_1->mat_elt    = ptr_2->mat_elt - ptr_3->mat_elt;

    cpt = cpt+2;
    i++;
```

```
    }
  }

  /**** Calcul de l'estimation des effets ***/
  ptr_1 = Goto_elt_double (1,k+1, max_line, factor+2,first_perf);
  ptr_2 = Goto_elt_double (1,k, max_line, factor+2,first_perf);
  ptr_1->mat_elt = ptr_2->mat_elt / max_line;
  rep_div = (int) (max_line / 2);
  for (i=2; i<=max_line; i++)
  {
   ptr_1 = Goto_elt_double(i,k+1,first_perf);
   ptr_2 = Goto_elt_double(i,k,first_perf);
   ptr_1->mat_elt   = ptr_2->mat_elt / rep_div;
  }

}
```

Figure 5.8 Codes de l'algorithme de Yates.

Cette partie consacrée à l'implantation des modules aurait put être plus détaillée, mais nous préférons juste donner un aperçu de cette implantation puisqu'elle a été totalement détaillée dans le manuel d'implantation de SimAd-Voltaire (do Régo 1992).

5.3 Utilisation de SimAd-Voltaire

Pour utiliser SimAd-Voltaire, il doit être installé sur une station de travail utilisant le système UNIX comme système d'exploitation, station ayant accès aux librairies constituant X-WINDOWS et enfin accès au compilateur Voltaire. Après les phases d'installation du logiciel l'utilisateur sera en mesure de démarrer une expérimentation.

5.3.1 Démarrage de SimAd-Voltaire

Après avoir passé le stade de sa connection sur la station de travail, l'utilisateur devra taper la commande:

prg% simad &

pour démarrer SimAd-Voltaire (prg% est le prompt du système)

Cette interface est composée de deux parties:

- au sommet de la fenêtre nous avons des boutons nous permettant d'activer différentes options
- une fenêtre de type texte chargée de recevoir le programme de simulation sous étude.

Les options mises à notre disposition sont les suivantes:

- "File" (Fichier)
- "Compile" (Compilation)
- "Input Parameters" (Paramètres d'entrées)
- "Analysis Type" (Type d'analyse)
- "Monitor" (Moniteur)
- "Advisor" (Conseiller)
- "Print" (Impression)
- "Options" (Options)
- "Quit" (Sortie)

L'utilisation de SimAd-Voltaire se fait normalement étape par étape en commençant par le bouton le plus à gauche et en allant vers celui se trouvant à l'extrême droite.

5.3.2 Le bouton "File" (Fichier)

Ce bouton permet d'effectuer le chargement d'un programme Voltaire se trouvant dans le répertoire de travail. Après avoir choisi l'option Load en cliquant sur le bouton gauche de la souris, l'utilisateur verra s'ouvrir une fenêtre lui présentant la liste des fichiers Voltaire se trouvant dans le répertoire (Figure 5.9). La sélection d'un fichier suivi d'une sélection du bouton Open, effectuera l'ouverture du fichier dans la fenêtre texte (Figure 5.10).

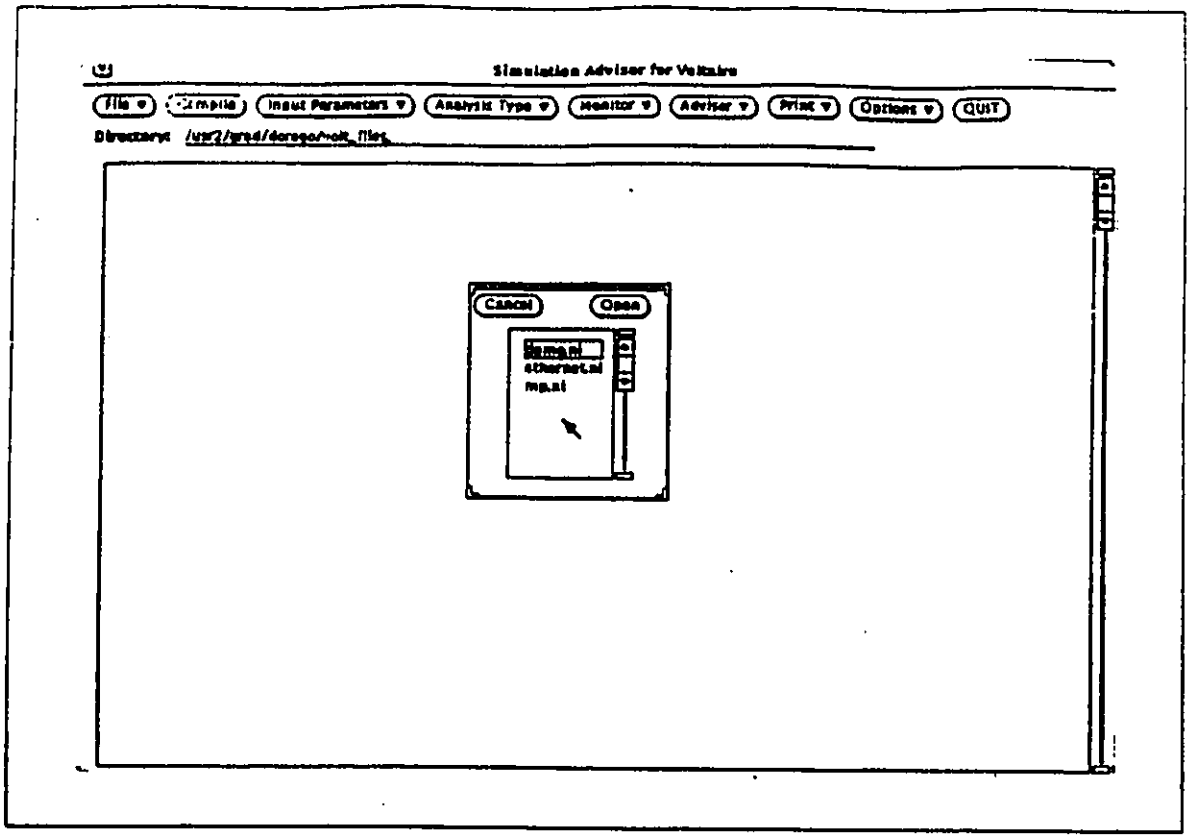

Figure 5.9 Liste des fichiers Voltaire

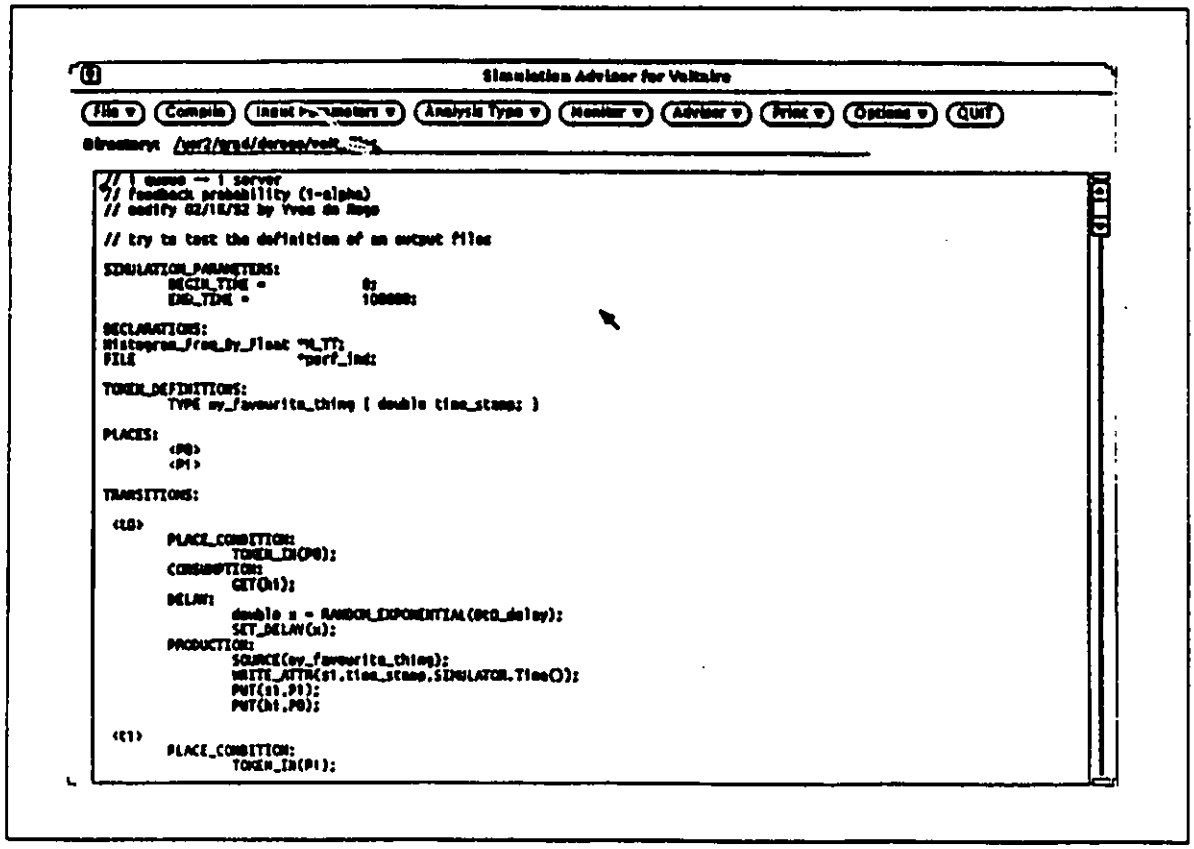

Figure 5.10 Ouverture du fichier demo.nl

5.3.3 Le bouton "Compile" (Compilation)

Ce bouton permettra d'effectuer la compilation du modèle de simulation, si la version compilée du programme n'existe pas. Mais nous conseillons plutôt l'utilisation de l'environnement du langage Voltaire. Cet environnement peut-être sélectionné à partir de SimAd-Voltaire en choisissant l'option Xrunner du bouton Options.

5.3.4 Structure d'un programme Voltaire à être traité par SimAd-Voltaire

Pour pouvoir conduire ses expérimentations l'utilisateur devra ajouter quelques lignes à ses programmes Voltaire. Ces lignes sont celles indiquées en gras dans le programme Voltaire présenté dans la Figure 5.11.

```
// 1 queue -- 1 server
// feedback probability (1-alpha)
// modify 02/16/92 by Yves do Rego

// try to test the définition of an output files

SIMULATION_PARAMETERS:
          BEGIN_TIME =                0;
          END_TIME =                  100000;

DECLARATIONS:
Histogram_Freq_By_Float *H_TT;
FILE                    *perf_ind;          /* Pour SimAd */

TOKEN_définitionS:
          TYPE my_favourite_thing { double time_stamp; }

PLACES:
          <P0>
          <P1>

TRANSITIONS:

 <t0>
          PLACE_CONDITION:
                    TOKEN_IN(P0);
          CONSUMPTION:
                    GET(h1);
          DELAY:
```

```
                double x = RANDOM_EXPONENTIAL(@t0_delay);
                SET_DELAY(x);
        PRODUCTION:
                SOURCE(my_favourite_thing);
                WRITE_ATTR(s1,time_stamp,SIMULATOR.Time());
                PUT(s1,P1);
                PUT(h1,P0);

<t1>
        PLACE_CONDITION:
                TOKEN_IN(P1);
        CONSUMPTION:
                GET(h1);
        DELAY:
                double x = RANDOM_EXPONENTIAL(@t1_delay);
                SET_DELAY(x);
        PRODUCTION:
                double x = RANDOM_UNIFORM(0,1);
                if (x < @alpha)
                 { if (SIMULATOR.Time() > 0)
                    { H_TT->Insert(
                          SIMULATOR.Time() -
READ_ATTR(h1,time_stamp) );
                    }
                   SINK(h1);
                 }
                else
                 { PUT(h1,P1); }


INITIAL_MARKING:
TYPE=my_favourite_thing,PLACE=P0,COUNT=1;

BEGIN_SIMULATION:


        perf_ind = fopen("PERFORM.DAT","w"); /* Pour  SimAd */

printf("This is the beginning of a simulation running from %g to %g\n",
           SIMULATOR.Start_Time(),SIMULATOR.End_Time());
         printf("statistics starting at %g\n",0);
         H_TT = new Histogram_Freq_By_Float(0,20,50,"Total time through
system","freq","time");


END_SIMULATION:
H_TT->Save("lq1s");
```

```
// compute the theoretical mean delay
double lambda, mu, loop_delay, total_delay;
lambda = 1. / @t0_delay;
mu = 1. / @t1_delay;
if (mu < lambda/@alpha)
{ printf("this is not a stable case: mu = %f is less than lambda/alpha =
%f\n",mu,lambda/@alpha);
   fprintf(perf_ind,"%f\n", -9999);          /* Pour  SimAd */
 }
else
 { loop_delay = 1. / (mu - lambda/@alpha);
   total_delay = loop_delay / @alpha;
   printf("Theoretical mean total delay = %f\n",total_delay);
   fprintf(perf_ind,"%f\n",total_delay);     /* Pour  SimAd */
 }
   fclose (perf_ind);                        /* Pour  SimAd */
```

Figure 5.11 Programme Voltaire: Une file d'attente-un serveur
(Ajout des lignes en gras)

Comme nous l'avons introduit dans la section 5.2.5, l'utilisateur doit définir un fichier nommé PERFORM.DAT dans lequel il rangera la valeur numérique de l'indice de performance obtenu après exécution du modèle. Les ajouts au programme pour qu'il puisse être traité par SimAd-Voltaire sont expliqués dans la figure 5.12.

`FILE *perf_ind`	création d'une variable de type fichier
`perf_ind = fopen("PERFORM.DAT","w")`	ouverture du fichier PERFORM.DAT
`fprintf(perf_ind,"%f\n", -9999)`	signifie pas de réponses logique
`fprintf(perf_ind,"%f\n",total_delay)`	réponse du système
`fclose (perf_ind)`	ferméture du fichier PERFORM.DAT

Figure 5.12 Explication des ajouts au programme Voltaire de la figure 5.11

5.4 Portée du logiciel

Le logiciel SimAd-Voltaire décrit dans cette thèse est opérationnelle. Nous avons pu conduire des expérimentations complète sur différents modèles de simulation. La portée de SimAd-Voltaire est déterminée par celle du langage Voltaire. Pour cela le lecteur pourra se référer à la section 4.2.1 du document.

Chapitre 6 - Résultat expérimental

Nous présentons dans ce chapitre un exemples de modèle de simulation montrant d'une part la puissance du langage Voltaire mais aussi cell de la méthode d'expérimentation factorielle par l'intermédiaire du logiciel SimAd-Voltaire. L'exemple choisi provient du tutoriel du langage Voltaire, il consistera en la modélisation d'un système ayant une file d'attente et un serveur avec retour (feedback).

6.1 Modèle: Une file d'attente d'un serveur avec retour (feedback)

Nous présentons dans cette liste un modèle très simple de file d'attente avec une boucle de retour (feedback). Le modèle est représenté dans la Figure 6.1.

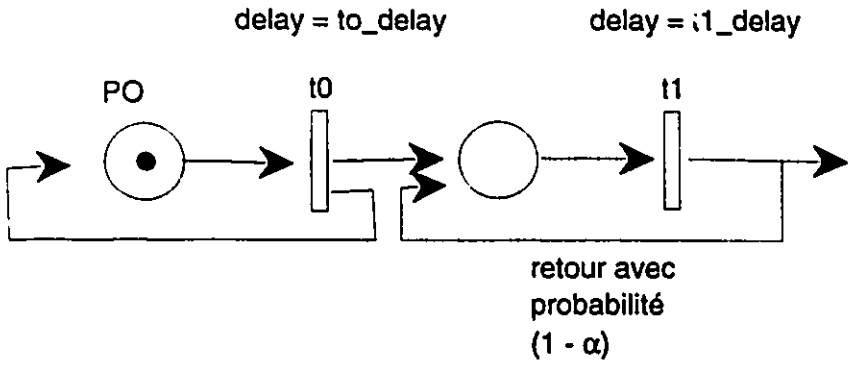

Figure 6.1 Système de file d'attente avec un serveur. La place P0 et la transition t0 contrôle l'arrivée des évènements dont le taux est contrôlé par le paramètre t0_delay. La place P1 et la transition t1 contrôle la file d'attente dont le taux est contrôlé par le paramètre t1_delay.

Le temps d'arrivée des clients dans ce système de file d'attente est déterminé par le paramètre t0_delay, qui est la moyenne des temps d'arrivée exponentiellement distribué. Le taux d'arrivée est donc $\lambda = 1/t0\_delay$.

Le temps de service est aussi exponentiellement distribué $\mu = 1/t1\_delay$.

Quand il n'y a pas de retour le temps d'attente dans le système peut être déterminé à partir de la formule:

$$E(t) = \frac{1}{\mu - \lambda}$$

Ce temps devient instable aussitôt que λ est plus grand que μ parce que le taux d'arrivée des clients devient plus grand que la capacité du serveur. Quand les taux sont justes égaux il n'y a pas d'équilibre non plus. C'est seulement quand le taux d'arrivée est strictement plus petit que le temps de service qu'il y a une solution. Quand la probabilité de retour est de $(1 - \alpha)$ le temps dans le système est évalué par la formule:

$$E(t) = \frac{1}{\alpha * \mu - \lambda}$$

La description de ce modèle en langage Voltaire est donnée dans la figure 6.2.

```
// 1 queue -- 1 server
// feedback probability (1-alpha)
// modify 02/16/92 by Yves do Rego

// try to test the définition of an output files

SIMULATION_PARAMETERS:
        BEGIN_TIME =                0;
        END_TIME =                  100000;

DECLARATIONS:
Histogram_Freq_By_Float *H_TT;
FILE                    *perf_ind;    /* Pour SimAd */

TOKEN_définitionS:
```

```
        TYPE my_favourite_thing { double time_stamp; }

PLACES:
        <P0>
        <P1>

TRANSITIONS:

 <t0>
        PLACE_CONDITION:
                TOKEN_IN(P0);
        CONSUMPTION:
                GET(h1);
        DELAY:
                double x = RANDOM_EXPONENTIAL(@t0_delay);
                SET_DELAY(x);
        PRODUCTION:
                SOURCE(my_favourite_thing);
                WRITE_ATTR(s1,time_stamp,SIMULATOR.Time());
                PUT(s1,P1);
                PUT(h1,P0);

 <t1>
        PLACE_CONDITION:
                TOKEN_IN(P1);
        CONSUMPTION:
                GET(h1);
        DELAY:
                double x = RANDOM_EXPONENTIAL(@t1_delay);
                SET_DELAY(x);
        PRODUCTION:
                double x = RANDOM_UNIFORM(0,1);
                if (x < @alpha)
                 { if (SIMULATOR.Time() > 0)
                    { H_TT->Insert(
                          SIMULATOR.Time() -
READ_ATTR(h1,time_stamp) );
                    }
                   SINK(h1);
                 }
                else
                 { PUT(h1,P1); }


INITIAL_MARKING:
TYPE=my_favourite_thing,PLACE=P0,COUNT=1;
```

```
BEGIN_SIMULATION:

        printf("Opening of the perf_index file\n");  /* Pour  SimAd */

        perf_ind = fopen("PERFORM.DAT","w");         /* Pour  SimAd */

printf("This is the beginning of a simulation running from %g to %g\n",
            SIMULATOR.Start_Time(),SIMULATOR.End_Time());
          printf("statistics starting at %g\n",0);
          H_TT = new Histogram_Freq_By_Float(0,20,50,"Total time through
system","freq","time");


END_SIMULATION:
H_TT->Save("lqls");
// compute the theoretical mean delay
double lambda, mu, loop_delay, total_delay;
lambda = 1. / @t0_delay;
mu = 1. / @t1_delay;
if (mu < lambda/@alpha)
{ printf("this is not a stable case: mu = %f is less than lambda/alpha =
%f\n",mu,lambda/@alpha);
  fprintf(perf_ind,"-9999.99");                      /* Pour  SimAd */
   fclose (perf_ind);                                /* Pour  SimAd */
 }
else
 { loop_delay = 1. / (mu - lambda/@alpha);
   total_delay = loop_delay / @alpha;
   printf("Theoretical mean total delay = %f\n",total_delay);
   fprintf(perf_ind,"%f\n",total_delay);             /* Pour  SimAd */
   fclose (perf_ind);                                /* Pour  SimAd */

 }
```

Figure 6.2 Programme Voltaire: Une file d'attente avec un serveur

6.2 Analyse des effets avec SimAd-Voltaire

Pour ce modèle simplifié d'un système ayant une file d'attente et un serveur, nous pouvons considérer qu'un analyste serait intéressé à connaître l'impact des certains paramètres sur la réponse du système (l'indice de performance). Dans notre cas les variables sous études seront les suivantes:

- t0_delay qui détermine le temps d'arrivée des clients (jetons)
- t1_delay qui détermine le temps de service d'un client (jetons)
- alpha qui permet le calcul de la probabilité de retour.

Nous avons 3 variables chacune à 2 niveaux, qui définissent un design factoriel de 2^3 expériences.

La première étape à effectuer est celle de la spécification des valeurs des paramètres sous études, ceux-ci auront les valeurs défini dans la Table 6.1

Table 6.1 Spécification des niveaux des variables

Nom des variables	*Niveau bas*	*Niveau haut*
t0_delay	5	6
t1_delay	3	4
alpha	1	2

Les Figures 6.3 et 6.4 nous montrent les fenêtres utilisées dans SimAd-Voltaire pour la spécification des niveaux des variables.

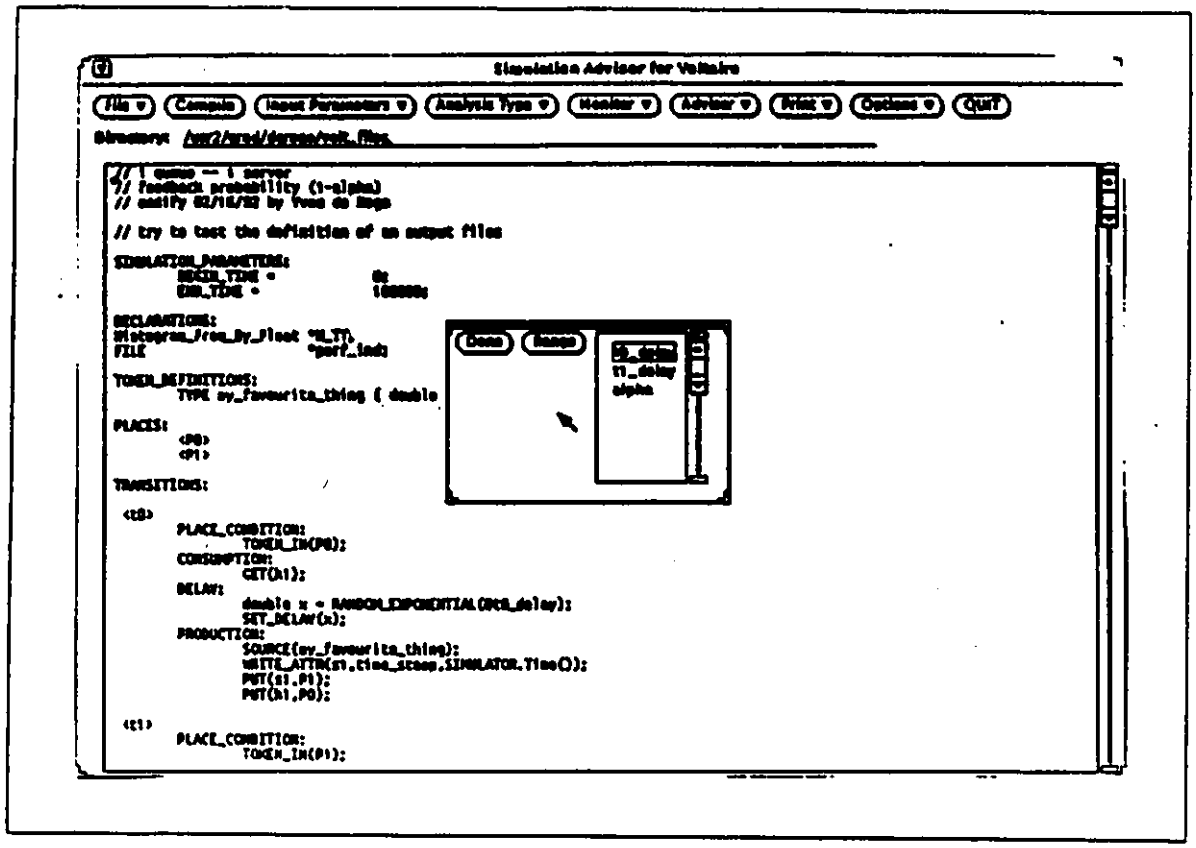

Figure 6.3 Sélection des variables (option Input Parameters)

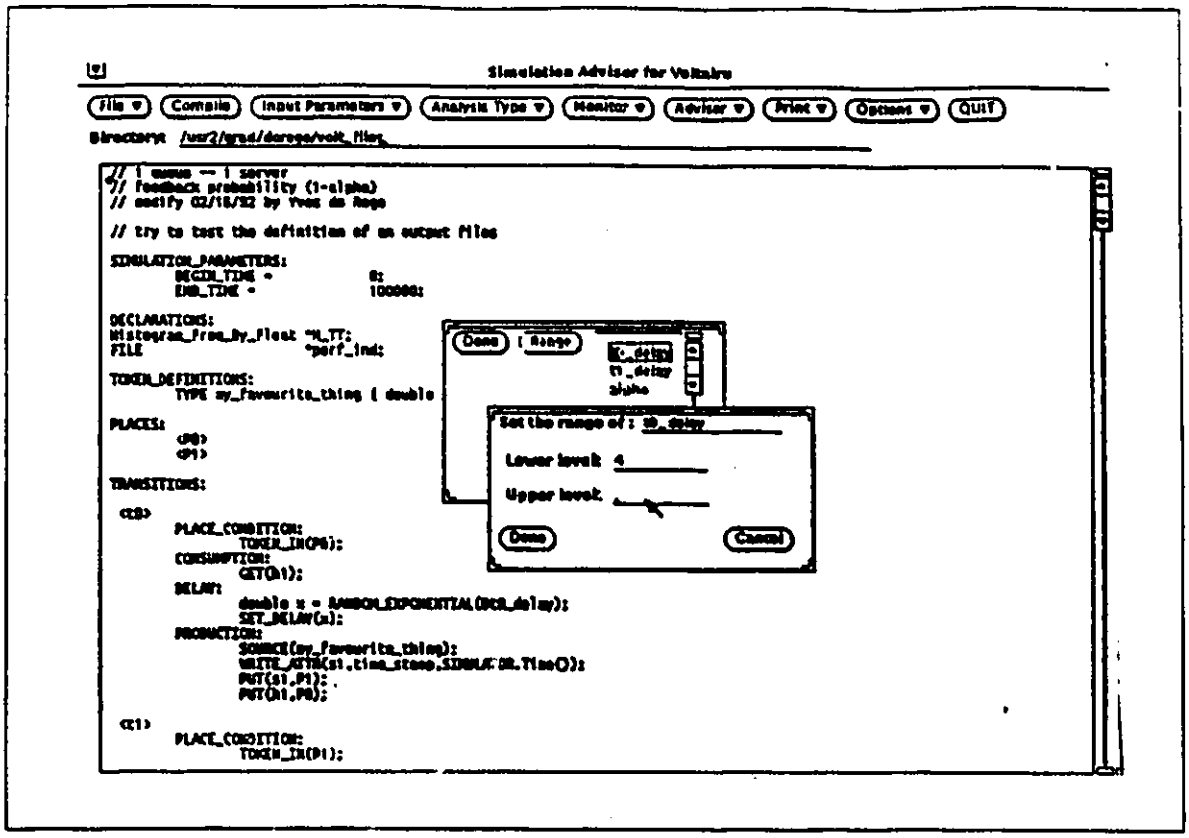

Figure 6.4 Spécification des valeurs des variables

La matrice de design générée par SimAd-Voltaire aura la structure et les informations indiquées dans la table 6.2. Elle présente les valeurs d'initialisation de chaque paramètre de décision pour chaque exécution du modèle.

SimAd-Voltaire offre deux modes d'exécution des 2^3 expériences: un selectif dans lequel l'analyste peut choisir l'expérience qu'il désire exécuter. Et un mode automatique qui exécute toutes les expériences sans intervention de l'analyste. Nous rappelons qu'après chaque exécution l'indice de performance ou la réponse du système est incorporé à la matrice de design.

Table 6.2 Matrice de design du modèle (une file d'attente avec un serveur)

Variables / Executions	*t0_delay*	*t1_delay*	*alpha*
1	5	3	1
2	6	3	1
3	5	4	1
4	6	4	1
5	5	3	2
6	6	3	2
7	5	4	2
8	6	4	3

Table 6.3 Matrice de design avec indices de performance

Variables / Exécutions	*t0_delay*	*t1_delay*	*alpha*	Réponses
1	5	3	1	7,5
2	6	3	1	6,0
3	5	4	1	20,0
4	6	4	1	12,0
5	5	3	2	2,14
6	6	3	2	2,0
7	5	4	2	3,33
8	6	4	2	3,0

Une fois les 2^3 expériences effectuées, l'analyste pourra choisir l'option qui lui permet de visualiser ou d'imprimer les effets des paramètres et leurs intéractions. Les effets seront représentés dans une table équivalente à celle de la table 6.4

Table 6.4 Effets de premier ordre et interactions du modèle d'une file d'attente avec un serveur

Effets de premier-ordre et effets issus d'interactions	
Moyenne	6,99702
A	-2,49405
B	5,17262
AB	-1,67262
C	-8,75595
AC	2,25595
BC	-4,07738
ABC	1,57738

Cette table représente les deux dernières colonnes d'une table de Yates, avant d'interprèter les informations de cette table nous devons rappeler que:

A représente *t0_delay*, le temps d'arrivée des clients,

B représente *t1_delay* le temps de service des clients et

C représente α nécéssaire au calcul de la probabilité de retour.

En reprenant les valeurs absolues les plus grandes nous pouvons distinguer les effets des facteurs ou interactions suivantes:

B, C et BC.

Ceci permet de dire que le temps de service attribué à chaque client à tendance à augmenter la réponse du systéme de 5,17, la probabilité de retour quant à elle a pour effet la diminution de la réponse du systéme. Simad-Voltaire présentera les résultats en utilisant les fenêtres présentées dans les figures 6.5 et 6.6.

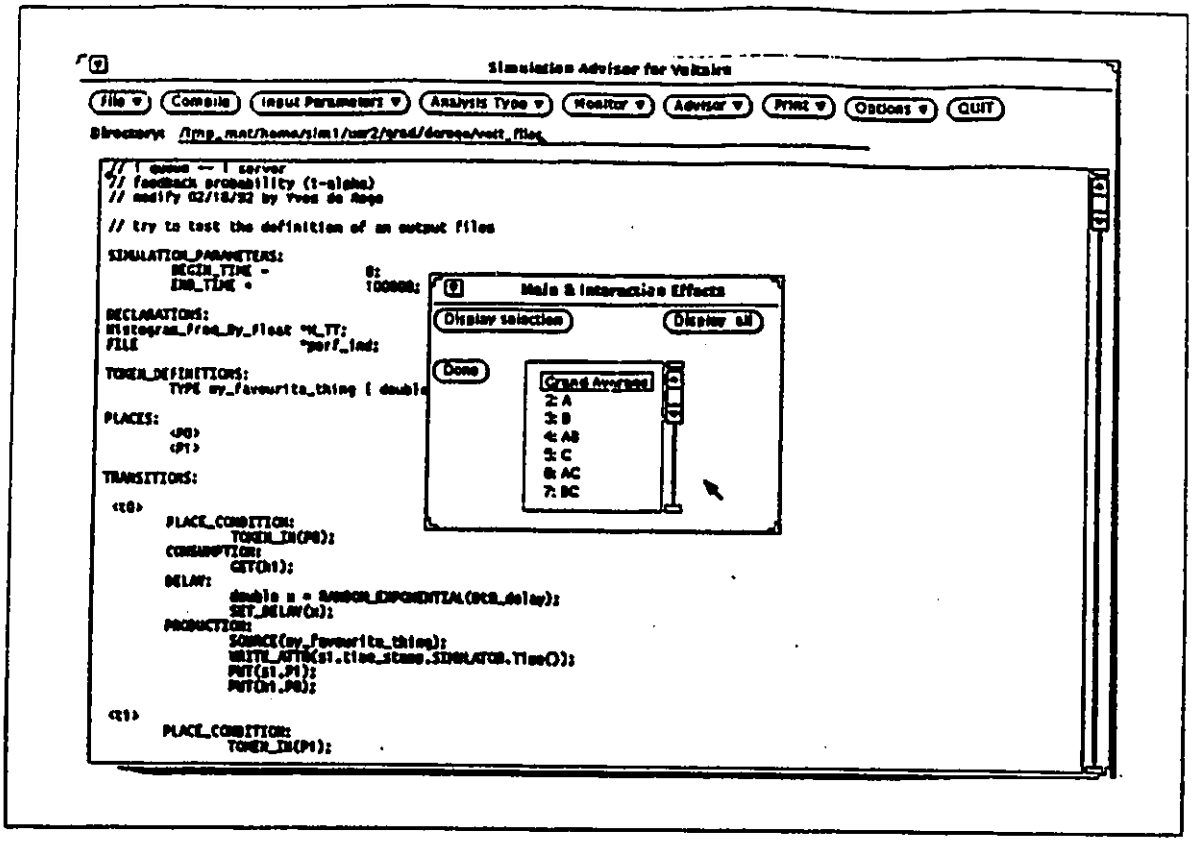

Figure 6.5 Effets de premier ordre et interactions (sélection)

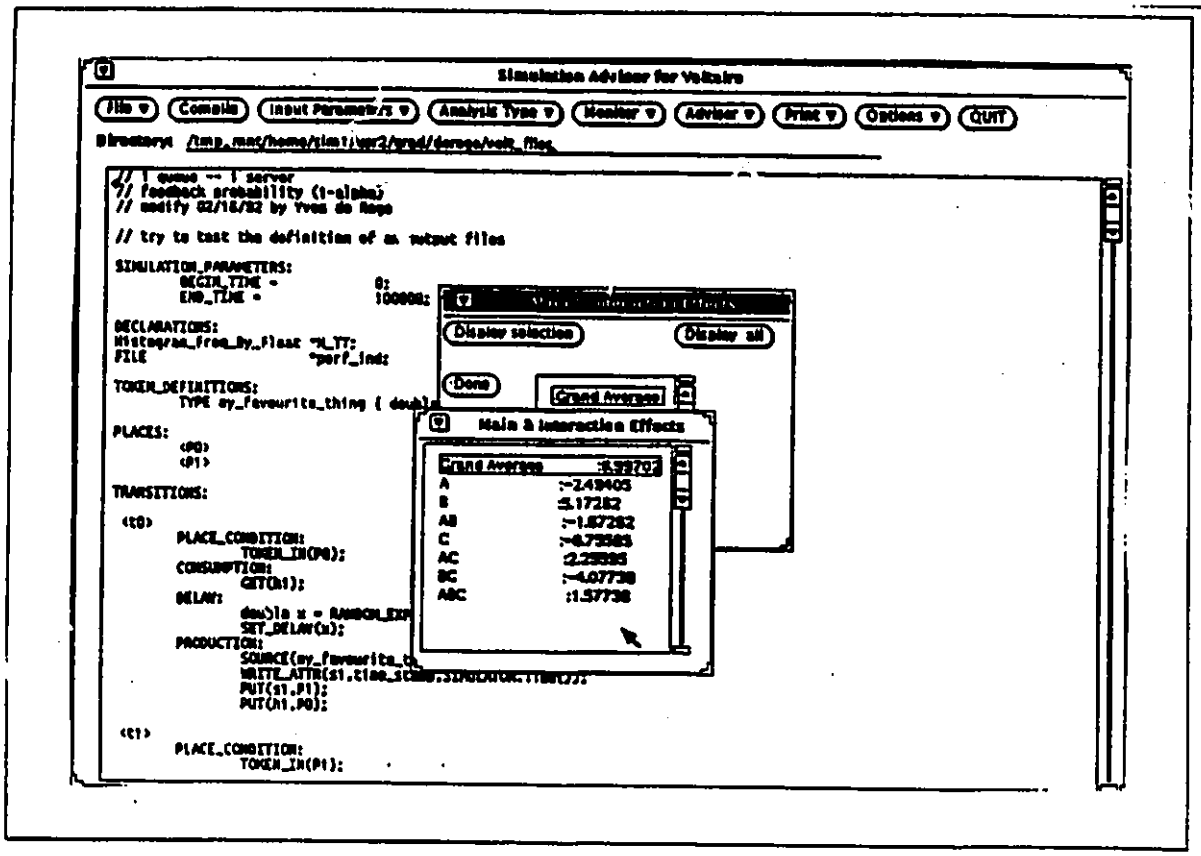

Figure 6.6 Effets de premier ordre et interactions (résultats)

Chapitre 7 - Conclusions et extensions futures

7.1 Commentaires sur les résultats

Après avoir présenté quelques méthodes permettant la conduite et l'analyse d'expérimentation, deux chapitres ont été consacrés aux méthodes d'analyse factorielle et fractionnelle factorielle. Ces méthodes simples au niveau de leur théorie s'avèrent très efficaces quant à la recherche des effets que peuvent avoir certaines variables sur la réponse d'un modèle de simulation. Cette thèse s'est plus particulièrement intéressée aux designs factoriels et fractionnels factoriels n'autorisant que deux niveaux par variables. Ce fait qui peut sembler restrictif ne l'est pas vraiment puisque des variables ayant plusieurs niveaux peuvent grâce aux combinaisons être traitées comme ayant deux niveaux.

La méthode d'analyse factorielle qui est la base de ce document s'avèrent très coûteuse en terme d'expériences à réaliser quand le nombre de variables sous étude devient important. En effet la croissance de ce nombre d'expériences est exponentielle puisqu'elle est déterminé par la fonction $2^{(\text{nombre de variables sous étude})}$. L'analyse fractionnelle fractorielle a été définie pour pallier à ce problème. Les bases de cette dernière sont les redondances existant entre les expériences réalisées sur le modèle. En effet celles-ci entraînent une réévaluation systématique des effets de premier-ordre et des effets issus des interactions entre les variables. Pour éviter ces réévaluations coûteuses, la méthode fractionnelle analyse le mouèle en ne considérant qu'un sous-ensemble des $2^{(\text{nombre de variables sous étude})}$ expériences à réaliser. Cette méthode s'est avérée aussi efficace que la première puisque les estimations obtenues sont très bonnes en considérant les exécutions que l'analyste n'a pas eu à évaluer. Nous n'avons présenté dans ce document que la méthode permettant la génération d'un sous-ensemble d'expériences égale à la moitié des celles à conduire lors d'une expérimentation factorielle complète, parce que la génération de fraction plus petite obéit aux mêmes règles de conctructions.

Par rapport à un design factoriel complet de $2^{(\text{nombre de variables sous étude})}$, le design fractionnel présenté est composé de $2^{(\text{nombre de variables sous étude})-1}$ expériences. Cette méthode fractionnelle factorielle a aussi des limitations au niveau de la fractionnalisation des expériences qui entraine des pertes d'informations qui deviennent importantes quand les fractions deviennent plus petites.

Le logiciel SimAd-Voltaire permet à l'analyste d'effectuer automatiquement les étapes concernant la construction du design, l'exécution des expériences, de même que l'analyse des effets des variables sous études. Nous rappelons que ces effets ne peuvent être interprètés que dans le contexte de leur exécution, donc l'analyste utilisant SimAd-Voltaire ne devra pas tirer de conclusions trop général concernant les effets car ceux-ci ne s'appliquent qu'aux points de design (niveaux de chaque variables).

Et puisque nous sommes dans le domaine de la simulation, l'analyste devra aussi tenir compte des effets des paramètres cachés, comme par exemple l'effet des valeurs initiales des générateurs de nombre pseudo-aléatoire utilisés.

7.2 Développements futurs de SimAd

Le logiciel SimAd-Voltaire que nous avons présenté dans le chapitre 5 de cette thèse permet la conduite d'expérimentations obéissant aux règles des designs factoriels et fractionnels factoriels. Celui-ci conçu de façon modulaire demeure évolutif en ce sens que plusieurs fonctionnalités pourront se greffer autour du système présentement opérationnel. Ce dernier est opérationnel au niveau de l'expérimentation factorielle de k variables et fractionnelle factorielle de k-p variables. Nous rappelons qu'une expérimentation va de la génération des expériences jusqu'aux calculs des effets.

Les développement futurs de l'architecture SimAd pourraient se situer dans les domaines suivants:

* Traitement d'autre langage de simulation.
* Implantation de nouvelles méthodes d'expérimentation.
* Implantation de différentes méthodes d'analyse.

Mais tous ces développements ne pourraient être efficaces que si le nombre de variables ou paramètres de décisions du modèle permettent une analyse factorielle ou fractionnelle peu coûteuse en terme de nombre d'expériences à conduire. Or il se trouve que la majorité des systèmes que l'on cherche à simuler font intervenir des dizaines voir des centaines de paramètres de décisions. Il serait donc bon d'avoir un module permettant avant la conduite de l'expérimentation la recherche des variables importantes du modèle de simulation. Pour ce faire nous présentons dans la section suivante une méthode qui serait applicable à l'architecture SimAd.

7.3 Comment identifier les variables importantes dans un programme de simulation

La méthode de recherche présentée par Bettonvil et Kleijnen (1991) traite le modèle de simulation comme une boîte noire, cette méthode comme celle d'analyse factorielle et fractionnelle factorielle a l'avantage d'être simple, mais elle ne permet pas d'étudier la structure interne du modèle. Les bases de cette méthode sont les suivantes:

Un modèle de simulation peut-être représenté mathématiquement comme:

$$y = s(v_1, \ldots, v_j, \ldots, v_N, r)$$

où s( ) représente la fonction mathématique définissant le programme de simulation; v_j le $j^{ième}$ facteur ou paramètre de décision. N est un entier positif connu; r est la valeur initiale du générateur de nombre pseudo-aléatoire; et y est la réponse du programme de

simulation. Comme dans la méthode factorielle l'utilisateur doit définir deux niveaux par facteur, et définir aussi l'indice de performance du système. C'est-à-dire que la réponse du programme de simulation devra obligatoirement être numérique. La méthode de recherche est séquentielle en ce sens que l'évaluation d'une expérience dépend des résultats de celle qui la précède. Une autre hypothèse est l'absence d'effets négatifs de premier-ordre et la croissance des réponses au fur et à mesure que j augmente; c'est-à-dire que $y_{(0)}$ est supposé être inférieur à $y_{(1)}$ lui même inférieur à $y_{(2)}$ etc... $y_{(j)}$ désigne la réponse du système quand les facteurs #1 à #j sont à leur niveau haut et les autres à leur niveau bas. La recherche s'effectue en plusieurs étapes d'évaluation du modèle:

étape 0 : Evaluation de $y_{(0)}$ et $y_{(N)}$
si ($y(0) = y(N)$) alors aucun facteur n'est important
sinon aller au étape 1

étape 1 : si ($y_{(N)} > y_{(0)}$) alors on évalue $y_{(N/2)}$
si ($y_{(N/2)} > y_{(0)}$) alors la première moitié contient au moins un facteur important.
si ($y_{(N/2)} < y_{(N)}$) alors la seconde moitié contient au moins un facteur important.

étape s : à cette étape il y aura 2^s groupes, chacun composé de 2^{m-s} facteurs ($N = 2^m$).
Si l'on suppose que $y_{(j1)} < y_{(j2)}$ avec $j1 < j2$
alors le groupe correspondant contient au moins un facteur important et au stade s+1 on découpera le groupe en deux sous-groupes égaux.

étape s+1 : On met à leur niveau haut les facteurs de #1 à #j3 avec j3 = (j1 + j2)/2 (donc j3 est le dernier facteur du nouveau groupe); les autres facteurs sont à leur niveau bas alors y(j3) est comparé avec y(j1) et y(j2). Cette comparaison entrainera l'abandon d'un groupe. Le groupe restant sera découpé et évalué de la même façon, après m étapes les facteurs seront trouvés.

7.4 Conclusion

Nous espérons avoir pu montrer par l'intermédiaire de cette thése, une application utile des méthodes d'analyses et d'expérimentations factorielle et fractionnelle factorielle au niveau de la simulation sur ordinateurs. Cette application s'est concrétisée par le développement du logiciel SimAd-Voltaire. Ce logiciel comme nous l'avons présenté dans le chapitre 5 a été implanté en langage C sur une station Sparc. L'architecture SimAd qui est de conception modulaire demeure évolutif au niveau de l'ajout de différents languages de simulation et différentes méthodes d'expérimentation et d'analyse. Cette thèse, en nous permettant d'améliorer nos techniques de recherches en informatique, nous a aussi permis de développer et d'apprendre des méthodes de design et de développement de logiciels. Pour terminer nous aimerions encore une fois remercier le Docteur Tuncer Ören qui après nous avoir proposer ce sujet de thèse a su se rendre disponible pour nous permettre de finaliser ce travail.

Liste des références

Bettonvil, B. et Kleijnen, J. P. C. (1991). Identifying the Important Factors in Simulation Models with many Factors. Technical Report FEW 498, Tilburg University, Brabant, The Netherlands.

Birta, L. G. et So, M. (1990). NEMS: A Database Support Environment for Numerical Experimentation. Simulation, 54: 4 (Avril), 189-200.

Box, G. E. P., Hunter, W. J., Hunter, J. S. (1978). Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building. John Wiley, New York, 306 - 418.

Connor, W. S. et Zelen, M. (1959). Fractional Factorial Designs for Factors at Three Levels. National Bureau of Standards, Applied Mathematics Vol. 54, Washington, D.C.

do Régo, Y. (1992). Implementation Manual of SimAd-Voltaire. Technical Report Prepared for Bell Canada. Computer Science Department, University of Ottawa, Ottawa, Ontario.

Hunter, J. S. et al. (1988). Design and Analysis of Experiments. In: J.M. Juran and F.M. Gryna (eds.), Juran Quality Control Handbook, McGraw-Hill, New York, 26.1 - 26.81.

Jávor, A. (1986). Applications of Expert Sysytems Concepts to Adaptative Experimentation with Models. In: M.S. Elzas, T.I. Ören, and B.P. Zeigler (eds.), Modelling and Simulation Methodology in the Artificial Intelligence Era, Elsevier, Amsterdam, The Netherlands, 153 - 163.

Korn, G. A. (1989). Interactive Dynamic System Simulation. McGraw-Hill Book, New York.

Law, A. M. et Kelton, W.D. (1991). Simulation Modeling & Analysis. McGraw-Hill, New York, 656 - 695.

Mendenhall,W. et Sincich, T. (1984). Statistics for the Engineering and Computer Sciences. Dellen Publishing Company, San Francisco, California.

Mourant, R. R. (1986). Automated Statistical Analysis of Discrete Event Simulation. Northeastern University and Micro Simulation, Framigham, MA.

Neelamkavil F. (1987). Computer Simulation and Modelling. John Wiley, New York.

Parent, P. (1990). Voltaire Netlist Language Tuturial Manual. VLSI Design Laboratory, Department of Electrical Engineering, McGill University, Montreal, Québec.