INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

THÈSE de DOCTORAT

Spécialité INFORMATIQUE

présentée par

Service Commun de la Documentation
INPL
Nancy-Gratte-Ciel

Catherine MONGENET

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE NANCY

UNE MÉTHODE DE CONCEPTION D'ALGORITHMES SYSTOLIQUES, RÉSULTATS THÉORIQUES ET RÉALISATION

soutenue le 7 mai 1985, devant la commission d'examen :

D 136 039004 0

Présider ...sité de Nancy I, CRIN.

Examinateu... ...ersité Louis Pasteur de Strasbourg.

R. MOHR, INPL, CRIN.

G.R. PERRIN, Université de Franche-Comté.

P. QUINTON, IRISA.

136039 0040

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

CT 1985 MONGENET C.

THÈSE de DOCTORAT

Spécialité INFORMATIQUE

présentée par

Catherine MONGENET

Service Commun de la Documentation
INPL
Nancy-Brabois

UNE MÉTHODE DE CONCEPTION D'ALGORITHMES SYSTOLIQUES, RÉSULTATS THÉORIQUES ET RÉALISATION

soutenue le 7 mai 1985, devant la commission d'examen :

Président : J.P. FINANCE, Université de Nancy I, CRIN.

Examinateurs : J.F. DUFOURD, Université Louis Pasteur de Strasbourg.
R. MOHR, INPL, CRIN.
G.R. PERRIN, Université de Franche-Comté.
P. QUINTON, IRISA.

Je tiens à adresser mes plus vifs remerciements à,

J.P. FINANCE, mon directeur de thèse, d'avoir accepté de présider ce jury, pour l'intérêt qu'il a porté à ce travail et ses conseils judicieux,

G.R. PERRIN, de m'avoir proposé ce travail, pour son temps, son aide et nos discussions enrichissantes,

J.F. DUFOURD et R. MOHR, d'avoir accepté de faire partie du jury et d'être rapporteurs de cette thèse,

P. QUINTON, d'avoir examiné ce travail avec bienveillance et accepté de participer à ce jury.

J'adresse également de chaleureux remerciements,

aux membres du groupe COMÈTE du CRIN,

à M. LODAY-RICHAUD et R. SEROUL pour l'aide qu'ils m'ont apportée dans la formalisation mathématique de certains points de la méthode,

aux membres du Laboratoire de typographie informatique de l'Université Louis Pasteur de Strasbourg, où j'ai pu réaliser le logiciel, ainsi que la saisie de cette thèse,

M. KRETZ, J.J. PANSIOT, Y. ROY et R. STROSSER pour leur aide et leurs conseils,

D. FOATA et J. DÉSARMÉNIEN pour m'avoir appris à utiliser le formatteur de textes mathématiques TEX/SM 90, ainsi que le logiciel de prétraitement STRATEC, ce qui m'a permis d'obtenir un document aussi soigné,

L. TETTAMANTI pour avoir saisi une partie de cette thèse.

La composition de cet ouvrage a été réalisée par TEX au Laboratoire de typographie informatique de l'Université de Strasbourg.

SOMMAIRE

INTRODUCTION

CHAPITRE 1 : ARCHITECTURES PARALLÈLES

I. Les concepts de base du parallélisme

I.1. Le principe du pipeline
I.2. Les unités fonctionnelles
I.3. Les tableaux de processeurs
I.4. Les multiprocesseurs

II. Classification

II.1. La classification de FLYNN
II.2. La classification de HOCKNEY et JESSHOPE

III. Les ordinateurs spécialisés

IV. Parallélisme, algorithmique et langages

CHAPITRE 2 : LES ARCHITECTURES SYSTOLIQUES

I. Présentation

II. Un exemple de réseaux linéaires : le produit de convolution

III. Un exemple de réseaux bidimensionnels : le produit matriciel

IV. Conclusion

CHAPITRE 3 : PRÉSENTATION INTUITIVE DE LA MÉTHODE

I. Le domaine et la spécification du problème

II. L'espace-temps et la base canonique

III. Transformations appliquées à une représentation

IV. Réseaux systoliques associés à une représentation

CHAPITRE 4 : PRÉSENTATION FORMELLE DE LA MÉTHODE

I. Enoncé d'un problème systolisable

I.1. Forme de l'énoncé
I.2. Méthode de décomposition du problème

II Spécification et représentation d'un problème systolisable

II.1. Le domaine associé au problème
II.2. Spécification systolique du problème
II.3. Les vecteurs générateurs
II.4. Représentation d'un problème

III. Transformation appliquée à une représentation

IV. Relation de précédence de représentations

V. Détermination des réseaux associés à une représentation

V.1. Résultats mathématiques utilisés
V.2. Détermination de la fonction d'allocation associée à une direction $\vec{\xi}$
V.3. Création du réseau
V.3.1. Nature d'une suite de données
V.3.2. Orientation d'un chemin de données
V.3.3. Espacement des données sur les flots
V.3.4. Nombre de cellules de retard sur les flots
V.3.5. Ordre des données sur les flots
V.4. Exemples de détermination de réseaux
V.4.1. Le produit de convolution
V.4.2. Le produit matriciel

CHAPITRE 5 : PRÉSENTATION DU LOGICIEL SYSTOL

I. Les fonctions du logiciel

I.1. Saisie conversationnelle de la spécification systolique
I.2. Calcul de l'ensemble des représentations
I.3. Choix d'une direction d'allocation
I.4. Calcul des caractéristiques d'un réseau
I.5. Dessin et simulation d'exécution des réseaux

II. Organisation du logiciel

III. Limites actuelles

IV. Extensions prévues

V. Un exemple d'utilisation du logiciel

CHAPITRE 6 : AUTRES APPROCHES DE LA CONCEPTION D'ALGORITHMES SYSTOLIQUES

I. Critères de comparaison

II. Approche de MOLDOVAN

III. Approche de MIRANKER et WINKLER

IV. Approche de QUINTON

V. Comparaison des différentes approches

CONCLUSION

BIBLIOGRAPHIE

ANNEXE

Introduction

Les problèmes à caractère scientifique qui interviennent dans de nombreux domaines d'applications et de recherches tels que les problèmes de mécanique des fluides (en météorologie ou aérodynamisme), les problèmes de simulation (en astrophysique ou en recherche nucléaire), le traitement et la synthèse d'images, etc, sont caractérisés par la grande quantité de données numériques manipulées nécessitant de nombreuses opérations arithmétiques. De ce fait, les temps d'exécution sont bien souvent prohibitifs sur les machines séquentielles classiques, lorsqu'il s'agit d'un problème temps réel ou d'une simulation en temps "raisonnable".

Les progrès technologiques, en particulier au niveau des circuits VLSI (*cf.* [MEC 80]) ont permis et permettent encore des gains importants de vitesse d'exécution, mais ne pourront dépasser les limites imposées par les contraintes physiques des composants de circuits comme la vitesse de propagation du signal.

Aussi les informaticiens ont-ils orienté leurs recherches, simultanément aux travaux concernant l'électronique des circuits, vers une autre voie permettant d'augmenter la vitesse d'exécution, même sans augmenter la vitesse des composants individuels : le parallélisme.

Le principe de base du parallélisme est de permettre l'exécution, non plus d'une seule opération à un instant donné, comme sur les machines séquentielles, mais de plusieurs opérations simultanées.

Le parallélisme concerne de nombreux aspects de l'informatique : la conception de circuits, l'architecture des machines, l'algorithmique et la programmation. On trouve, entre autre, dans [INR 84] et [AFC 83] un aperçu de ces différents sujets.

Le développement du parallélisme est caractérisé par deux aspects. Le premier concerne la conception d'architectures parallèles. Il faut doter ces nouvelles architectures de moyens physiques permettant l'exécution de plusieurs actions simultanées. De nouvelles notions se sont développées dans ce contexte : pipeline, tableaux de processeurs élémentaires, SIMD, MIMD, cadencement par les données, etc, qui conduisent à des classes d'architectures parallèles différentes. Il faut également proposer des critères permettant de déterminer l'efficacité d'une classe d'architectures pour résoudre un problème donné (nombre opérations par seconde, temps d'exécution, nombre de processeurs simultanément actifs, etc).

Le deuxième aspect concerne la conception d'algorithmes parallèles : quelle méthode proposer pour passer d'un problème à un algorithme parallèle correspondant, exécutable sur une classe d'architectures parallèles, dans les meilleures conditions possibles, c'est-à-dire en utilisant au maximum le parallélisme? Comment interviennent les structures de données du problème? Quelle est l'influence de la classe d'architectures parallèles choisie sur la méthodologie?

Le travail présenté dans cette thèse se situe dans le cadre de la conception d'algorithmes parallèles, adaptée à une classe particulière d'architectures parallèles, proposées par H.T. KUNG (*cf.* [KUN 79], [KUN 82]), appelées ***les architectures (ou réseaux) systoliques.*** Ces réseaux sont des systèmes spécialisés caractérisés par une structure régulière de processeurs élémentaires identiques (ou de quelques types simples) connectés localement, à travers lesquels circulent les données de façon rythmée. Leur fonctionnement est synchrone : à chaque période de temps, un ensemble de calculs élémentaires est réalisé simultanément par les processeurs élémentaires, puis les données transitent vers un processeur voisin sur le chemin de communication, pour être utilisées lors de la période de temps suivante.

Pour ce type d'architectures, la mise en oeuvre d'un algorithme, dit ***algorithme systolique*** est fortement liée à l'architecture du réseau. L'algorithme s'exprime essentiellement par le ou les types de processeurs élémentaires utilisés et par les chemins de communication nécessaires. Ainsi, contrairement à la programmation classique où un algorithme quelconque peut s'exprimer sur une architecture simple de type Von Neuman en utilisant une structure de contrôle complexe, l'approche systolique consiste à concevoir une architecture complexe, constituée de nombreux processeurs élémentaires et d'un réseau de communication régulier, correspondant à un seul algorithme. La structure de contrôle est alors directement prise en compte dans l'architecture. Elle est donc réduite aux signaux de synchronisation.

L'essentiel de ce travail porte donc sur la conception d'algorithmes et de réseaux systoliques. Le problème que nous souhaitons résoudre peut s'exprimer ainsi : comment concevoir, à partir d'une spécification d'un problème, un ensemble d'algorithmes systoliques solution de ce problème.

L'objectif final est de proposer un logiciel qui, à partir d'une caractérisation d'un problème, détermine automatiquement un ensemble de réseaux systoliques intéressants réalisant les algorithmes systoliques solution du problème donné.

Après une description générale de différentes formes d'architectures parallèles au chapitre 1, le chapitre 2 présente les architectures systoliques et illustre leur utilisation sur deux exemples classiques : le produit de convolution et le produit matriciel. Au chapitre 3, nous analysons le problème posé et décrivons les points principaux de la méthode de conception d'algorithmes systoliques sur l'exemple du produit de convolution. Le chapitre 4 présente de manière formelle cette méthode.

Les principaux points de la méthode proposée sont :
- caractérisation systolique du problème à partir de son énoncé.
- représentation du problème par un réseau de points de coordonnées entières dans $\mathbf{R}^r$.
- application sur cette représentation d'une suite de transformations affines, définies en fonction de critères sur les données du problème. Cela permet d'obtenir d'autres représentations -ou ordonnancements temporels des calculs élémentaires- pour chacune desquelles on déduit un ensemble de réseaux systoliques.

Cette méthode permet de déterminer, à partir de l'énoncé d'un problème, un ensemble de réseaux systoliques associés au problème. D'autres approches de conception d'algorithmes et de réseaux systoliques ont été proposées par P.QUINTON [QUI 83], D.L. MOLDOVAN [MOL 83] et W.L. MIRANKER et A. WINKLER [MIW 82]. Notre méthode, comme celle de QUINTON et contrairement aux autres, est une méthode constructive donc implantable. On peut alors réaliser un logiciel correspondant qui permet d'obtenir les réseaux associés à un problème, de manière automatique, à partir de l'énoncé. Elle est pour l'instant limitée aux réseaux systoliques de forme classique proposés par KUNG (*cf.* [KUN 82]), mais une extension est possible pour obtenir des réseaux plus particuliers, comme les réseaux en anneaux (*cf.* [QUI 83]), s'il s'avère que de tels réseaux soient intéressants dans la pratique.

Le chapitre 5 présente SYSTOL, le logiciel mettant en oeuvre la méthode de conception proposée. Il a été réalisé en pascal sur SM90 et permet de visualiser les réseaux systoliques associés à un problème donné sur un écran bit-map. Au chapitre 6, nous comparons notre approche de conception d'algorithmes systoliques avec les approches proposés par P. QUINTON [QUI 83], D.J. MOLDOVAN [MOL 83] et W.L. MIRANKER et A. WINKLER [MIW 82].

CHAPITRE PREMIER

ARCHITECTURES PARALLÈLES

Les premiers ordinateurs construits au début des années 50 étaient basés sur le principe de séquentialité d'exécution proposé par Von Neumann. Ces machines comportaient une unité d'entrée-sortie, une mémoire pour le stockage à la fois des données et des instructions du programme à exécuter et une unité centrale (CPU) constituée d'une unité de contrôle pour l'interprétation des instructions et d'une unité arithmétique et logique, activées successivement. Leur principe de fonctionnement reposait sur le fait que les instructions du programme étaient exécutées séquentiellement l'une après l'autre, les opérations élémentaires les constituant (accès mémoire, opération arithmétique ou logique, calcul d'adresse, entrée-sortie, etc) étant elles-mêmes exécutées en séquence.

L'UNIVAC 1 construit sur ces principes et commercialisé en 1951, permettait d'exécuter une opération arithmétique simple en un temps d'environ une milliseconde ($10^{-3}s$).

Le CRAY-1, construit à la fin des années 70 [RUS 78] et basé sur des concepts de parallélisme, peut exécuter une opération en un temps avoisinant la nanoseconde ($10^{-9}s$).

L'augmentation de la vitesse d'exécution d'une opération en une trentaine d'années (facteur d'environ 10^6 entre l'UNIVAC 1 et le CRAY-1) est due à deux phénomènes :

— les progrès technologiques réalisés au niveau des composants matériels : les premiers tubes électroniques sont remplacés par les transistors, puis par les circuits intégrés et enfin actuellement par les circuits à très haute densité d'intégration ou circuits VLSI. Ces évolutions successives ont permis d'augmenter considérablement la miniaturisation, la fiabilité, la complexité et la vitesse des circuits. Ainsi à complexité sensiblement égale, la puissance, évaluée à un millier d'opérations par seconde il y a 30 ans, atteind maintenant les MFlops, c'est-à-dire million d'opérations par seconde.

— l'introduction du parallélisme à tous les niveaux de l'architecture. La première mise en oeuvre du parallélisme sur un monoprocesseur a été de permettre la simultanéité entre les calculs et les opérations d'entrée-sortie, augmentant ainsi le taux d'activité de l'unité centrale. De nouveaux concepts architecturaux ont permis d'introduire davantage de parallélisme et d'augmenter encore la vitesse d'exécution des opérations.

I. Concepts de base du parallélisme.

Une définition générale du parallélisme est de permettre l'exécution simultanée de différentes tâches. Sa mise en oeuvre peut prendre différentes formes basées sur quatre concepts différents décrits entre autre dans [HOJ 81] ou [ZAK 84] :
— la technique du pipeline
— l'introduction d'unités fonctionnelles dans un processeur
— les tableaux de processeurs
— les multiprocesseurs

I.1. Le principe du pipeline. — Le principe de la technique du pipeline s'apparente au fonctionnement d'une chaîne de montage, de voitures par exemple, où le travail à effectuer sur une voiture est morcelé en tâches. Une fois la chaîne "amorcée", toutes les tâches sont actives simultanément sur des voitures successives et une voiture sort terminée de la chaîne à intervalles réguliers.

Ce principe est appliqué à un opérateur décomposé en plusieurs tâches élémentaires, comme l'illustre la figure 1.1. On suppose que toutes les tâches ont la même durée t. Après un temps d'amorçage de n intervalles de temps, un résultat $R_i (i = 1, \ldots, p)$ est obtenu à chaque intervalle de temps. Les p résultats sont donc calculés en $n + p$ intervalles de temps. Ainsi une addition en virgule flottante peut être segmentée en 4 phases d'exécution successives : comparaison des exposants, normalisation de la

mantisse, addition des mantisses, normalisation du résultat. Chaque phase utilise les résultats de la phase précédente.

Si l'addition est à réaliser sur une longue chaîne de données, quatre additions sont simultanément exécutées à quatre phases différentes sur quatre paires d'opérandes successives. On parle alors de *recouvrement* d'exécution : l'exécution d'une instruction commence avant la fin de l'instruction précédente.

Le fonctionnement d'un opérateur pipeliné est donc particulièrement adapté aux traitements portant sur des données de nature vectorielle (vecteur, matrice ou suite), afin d'utiliser au maximum le parallélisme du au recouvrement partiel d'opérations successives de même type.

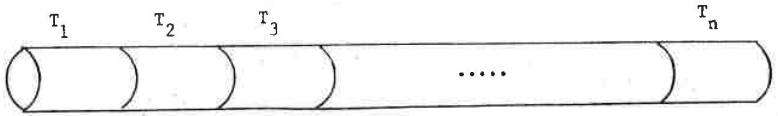
I.2. Les unités fonctionnelles. — Une autre manière d'augmenter la vitesse de calcul est d'utiliser, dans un processeur unique, plusieurs unités fonctionnelles indépendantes. Chaque unité peut être activée par une instruction séparée qui spécifie les données correspondantes (instruction logique, addition, multiplication, etc). Les différentes unités peuvent donc être activées simultanément : il y a alors recouvrement total d'exécution. Les fonctions peuvent être entièrement différentes, mais on peut concevoir aussi que certaines fonctions soient répétées, avoir par exemple deux additionneurs.

Un problème important lié au bon fonctionnement d'un tel système est celui de la *synchronisation* entre unités. Les différentes fonctions peuvent avoir des temps d'exécution et d'obtention de leurs opérandes différents. De plus, les opérandes d'entrée d'une fonction peuvent être les résultats d'une autre fonction.

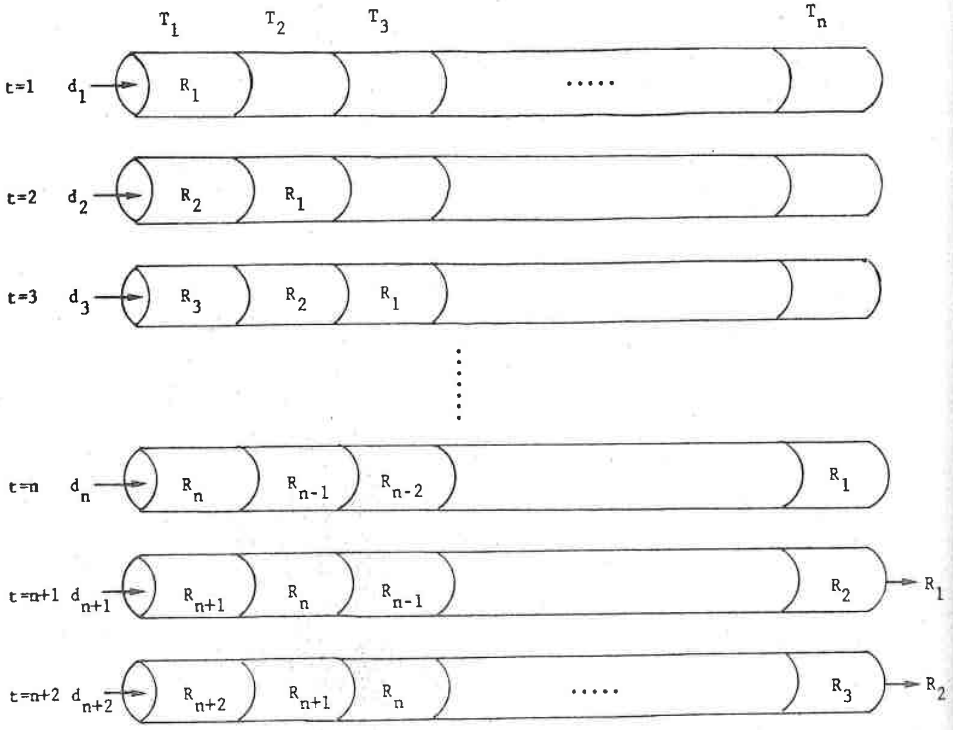
La tâche de programmation est donc très complexe. L'existence de bibliothèques de sous-programmes qui utilisent au maximum les possibilités de parallélisme offertes par les unités en traitant correctement les problèmes de synchronisation, est une aide précieuse pour le programmeur.

I.3. Les tableaux de processeurs (processor array). — Le principe est d'utiliser un ensemble de processeurs, appelés processeurs élémentaires (P.E.) identiques soumis au contrôle d'une seule unité de commande qui décode et diffuse les instructions à tous les processeurs du tableau. Ainsi, les processeurs exécutent de manière synchrone (simultanément) la même instruction sur des données différentes.

Dans ce type d'architecture, il faut accèder en parallèle aux différentes données utilisées par les processeurs élémentaires correspondants. Pour



(a) Décomposition de l'opération O en n tâches élémentaires successives



(b) Visualisation de l'opération pipelinée sur une suite de données vectorielles $(d_i)_{0 \leq i \leq p}$

FIGURE 1.1. Opérateur pipeliné

cela, il est nécessaire de *partitionner la mémoire en bancs*, les différents bancs étant accessibles simultanément par des processeurs différents.

La deuxième caractéristique de ce type d'architectures est le *réseau d'interconnexion* (*cf.* [LEN 82]). Il permet la communication entre processeurs ou entre processeurs et bancs mémoire. Différentes formes de réseaux sont proposés dans la littérature : réseau crossbar, réseau de Benes, etc. On trouve dans [HOJ 81] une présentation de ces différentes types de réseaux.

Une structure possible de ce type d'architecture est d'avoir autant de processeurs que de bancs mémoire. Un processeur est alors directement connecté à un seul banc. Le réseau d'interconnexion permet aux processeurs de communiquer entre eux (*cf.* figure 1.2). Il peut avoir une forme simple de connexion linéaire, un processeur communique avec ces deux processeurs voisins (nearest-neighbour network) (*cf.* figure 1.3(a)), ou de connexion bidimensionnelle, un processeur peut communiquer avec ses 4 ou 8 voisins (*cf.* figure 1.3(b)).

Lorsque le nombre de processeurs est différent du nombre de bancs mémoire, le réseau d'interconnexion doit relier les processeurs et les bancs mémoire (*cf.* figure 1.4), les processeurs pouvant à priori avoir accès à n'importe quel banc "disponible".

REMARQUE. — Le terme de calcul vectoriel (ou array processing) désigne l'ensemble des problèmes pouvant être exprimés algorithmiquement en termes d'opérations sur des tableaux réguliers de données. Cela concerne de nombreux problèmes numériques (résolution de systèmes linéaires, équations différentielles, récurrences, etc) utilisés dans des domaines scientifiques très variés (météorologie, séismologie, géophysique, etc).

Ces problèmes sont par nature fortement parallèles : les calculs portant sur différents éléments de tableaux peuvent être simultanés. De ce fait, ils tiennent une place importante dans le développement du parallélisme et ont donné lieu à la conception de nombreuses architectures parallèles.

Il faut remarquer que le terme d'"array processor" n'a pas toujours la même signification dans la littérature. Pour les uns (par exemple [HAY-al 82]), il désigne les architectures composées de processeurs identiques synchrones exécutant simultanément la même opération sur des données différentes. Cela correspond aux tableaux de processeurs décrits précédemment. Pour d'autres (par exemple [HOJ 81]), il désigne les architectures adaptées aux calculs sur les tableaux de données. Cela correspond non seulement aux architectures formées de tableaux de processeurs identiques

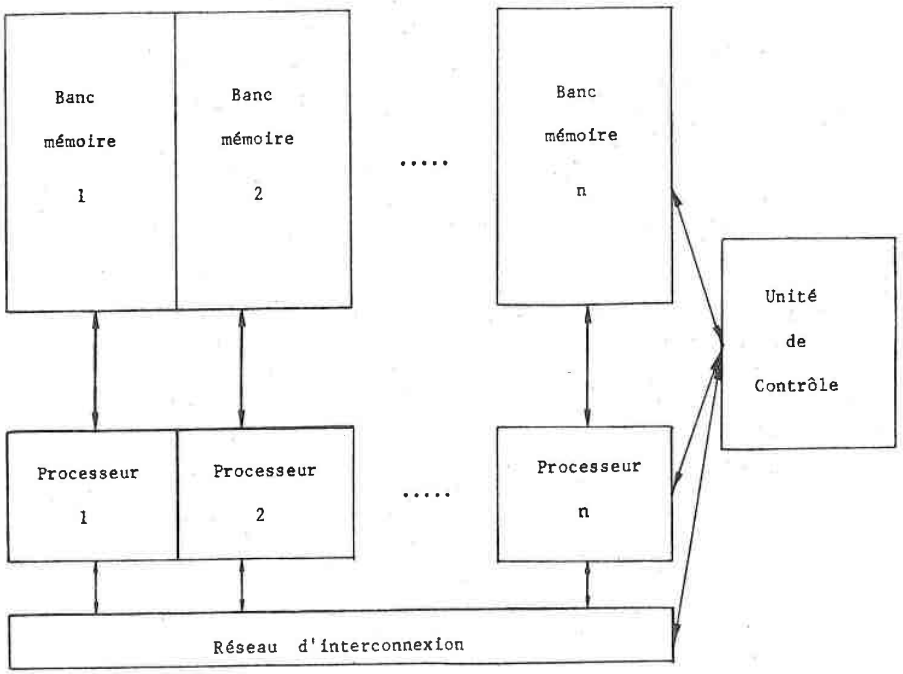


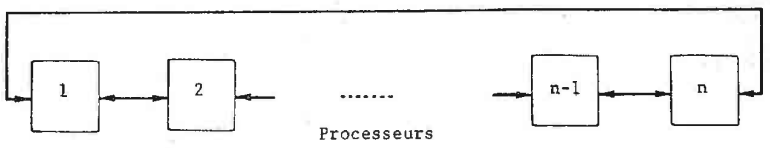
FIGURE 1.2. Une organisation de tableaux de processeurs

synchrones, mais aussi à d'autres architectures conçues à partir du principe du pipeline.

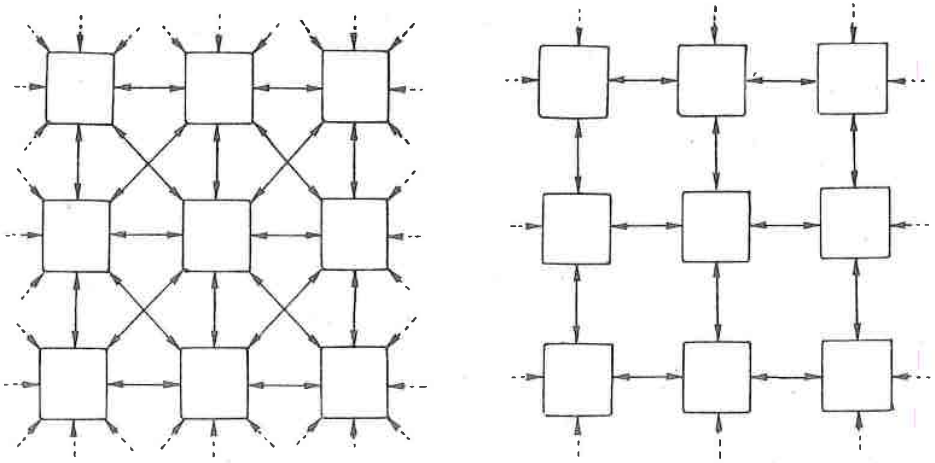
I.4. Les multiprocesseurs. — Contrairement à l'approche précédente, une architecture multiprocesseurs est composée de plusieurs processeurs asynchrones. Chaque processeur exécute ses propres instructions; les instructions exécutées simultanément peuvent être de types très différents.

Ce type d'architecture pose des problèmes de conflits d'accès aux bancs mémoire et de communication entre processeurs. Le contrôle est donc plus complexe ainsi que le réseau d'interconnexion.

De plus, les méthodes de conception d'algorithmes existantes, orientées vers le traitement séquentiel, ne permettent pas facilement l'utiliser au maximum la puissance des multiprocesseurs.



(a) Connexion linéaire



(b) Connexions bidimensionnelles

FIGURE 1.3. Exemples de réseaux de connexions de processeurs

Parmi les différentes architectures multiprocesseurs proposées dans la littérature, on distingue deux classes :

— les machines constituées d'un ensemble de processeurs séquentiels classiques, comme CMMP ou CM* (*cf.* [ENS 77]).

— les machines cadencées par les données (data-flow computers) (*cf.* [COM 82]) dont une réalisation est le système LAU (Langage à Assignation

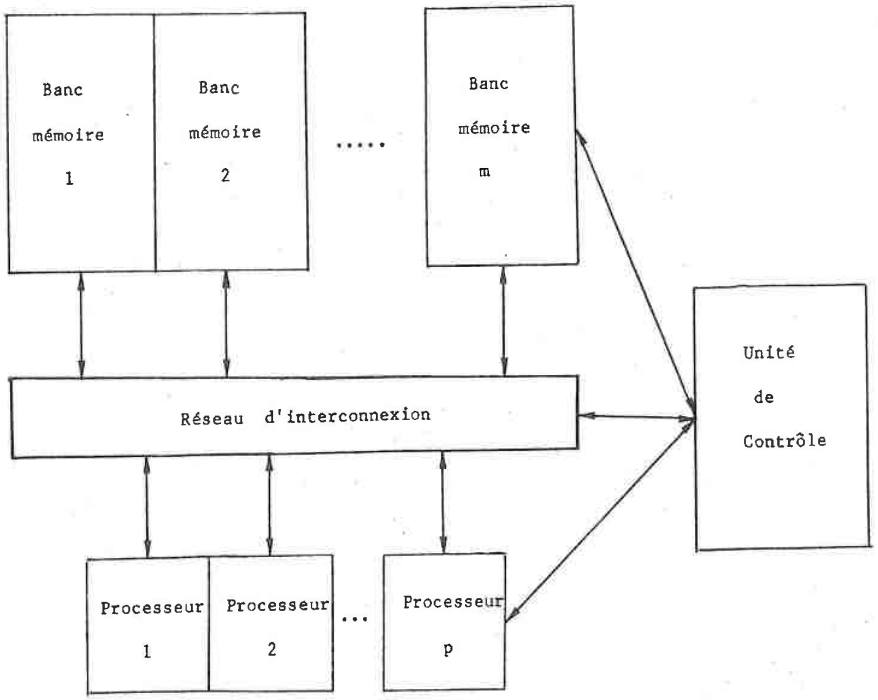


FIGURE 1.4. Une organisation de tableau de processeurs

Unique) (*cf.* [BER-al 81]). Ces architectures sont basées sur le principe suivant : une instruction est exécutée dès que ses opérandes ont été calculés. Ainsi quand plusieurs opérations sont exécutables, c'est-à-dire ont leurs opérandes disponibles, elles sont assignées à des processeurs différents et exécutées. L'exécution d'une instruction rend disponible un résultat et permet ainsi à d'autres instructions, utilisant ce résultat comme donnée, de devenir actives.

II. Classification

Les concepts de base décrits précédemment peuvent se combiner dans une même architecture. Ainsi, une unité fonctionnelle d'un ordinateur à plusieurs unités peut être un tableau de processeurs, ou les processeurs élémentaires d'un tableau de processeurs peuvent être des unités arithmétiques pipelinées.

Ces combinaisons de plusieurs concepts rend difficile une classification des différents types d'architectures parallèles. Plusieurs synthèses sur les architectures parallèles ont été proposées (*cf.* [HAY-al 82] [HOJ 81], [COS 83], [ZAK 84]), ainsi que des classifications dont nous retenons celles de M.J. FLYNN ([FLY 72]) et de R.W. HOCKNEY et C.R. JESSHOPE (*cf.* [HOJ 81]).

II.1. La classification de FLYNN [FLY 72]. — C'est la classification la plus simple et la plus connue. Elle repose, non pas sur la structure des machines parallèles, mais sur le fait que les suites d'instructions et de données dans un ordinateur peuvent être uniques ou multiples. Elle propose ainsi quatre classes d'architectures :

— SISD (Single Instruction stream/Single Data stream). Cette classe correspond à la machine séquentielle de type Von Neumann.
— SIMD (Single Instruction stream/Multiple Data stream). Elle correspond aux machines où une même instruction est exécutée sur un ensemble de données différentes. Dans cette classe se situent les architectures constituées de tableaux de processeurs (processor array).
— MISD (Multiple Instruction stream/Single Data stream). Cette classe, correspondant aux machines où plusieurs instructions travaillent sur une même donnée, semble artificielle.
— MIMD (Multiple Instruction stream/Multiple Data stream). Cette classe correspond à toutes les formes de multiprocesseurs.

Cette classification présente une ambiguité pour les architectures vectorielles pipelinées. On appelle architecture vectorielle pipelinée, une machine basée sur la technique du pipeline offrant en plus des instructions vectorielles (de traitement de vecteurs) comme le CRAY-1 (*cf.* [RUS 78]) ou le CDC CYBER 205 (*cf.* description dans [HOJ 81]).

Pour FLYNN, ces architectures appartiennent à la classe SIMD car il considère qu'une instruction vectorielle (Single Instruction Stream) opère

sur un ensemble de données, ici des éléments de vecteur (Multiple Data Stream). D'autres, dont HOCKNEY et JESSHOPE (*cf.* [HOJ 81]) préfèrent classer ces architectures dans la catégorie SISD car un calcul vectoriel s'effectue sur une suite de données vectorielles (Single Data Stream).

Il faut remarquer que contrairement à l'intention de FLYNN, les informaticiens ont adopté la convention implicite suivante :
— SIMD est synonyme de tableau de processeurs synchrones (processor array) avec une unité de contrôle unique. Une architecture de ce type est l'ILLIAC IV (*cf.* [BAR-al 68]).
— MIMD est synonyme de multiprocesseur, ensemble de processeurs communicants indépendants, comme dans la classification de FLYNN.

II.2. La classification de HOCKNEY et JESSHOPE [HOJ 81]. — En raison des ambiguités de la classification de FLYNN, HOCKNEY et JESSHOPE se sont orientés dans une toute autre voie et ont tenté de classer les différentes architectures, séquentielles et parallèles, en fonction de leur structure, en particulier selon la présence ou non des différentes formes de parallélisme : pipeline, parallélisme fonctionnel, répétition de processeurs, multiprocesseurs.

Pour cela, ils définissent une notation structurale permettant de décrire les caractéristiques des différentes classes d'architectures. Elle permet de définir, le nombre d'unités d'instructions (unité décodant une suite d'instructions), d'unités d'exécutions (ou unité arithmétique et logique) et d'unités de mémoire (nombre de bancs), leurs interconnexions et le contrôle.

Ils obtiennent ainsi une classification structurale beaucoup plus fine, en une quinzaine de classes, qui semble peu utilisable dans la pratique.

III. Les ordinateurs spécialisés

L'évolution rapide de la technologie des circuits intégrés, en particulier les niveaux d'intégration maintenant accessibles (*cf.* [MEC 80]), conduit à une diminution de coût et une augmentation du nombre et de la complexité des composants d'un circuit. De ce fait, on assiste à l'avènement de nombreux ordinateurs spécialisés adaptés à un type de problème donné.

Dans ce cadre, se sont développés de nombreux "array processors" au sens le plus large indiqué ci-dessus, principalement pour les problèmes de traitement du signal.

Les architectures systoliques proposées par H.T. KUNG (*cf.* [KUN 79], [KUN 82]), auxquelles nous nous intéressons exclusivement dans la suite, sont de telles architectures spécialisées. Elles permettent de résoudre des problèmes s'exprimant sous forme de récurrence et sont caractérisées par une structure régulière de processeurs élémentaires (ou cellules) connectés localement à travers lesquels circulent les données de façon rythmée. L'enchainement des calculs est pipeliné.

IV. Parallélisme, algorithmique et langage

Outre les problèmes matériels liés au parallélisme déjà évoqués précédemment : conflit d'accès à la mémoire, communication entre processeurs, réseaux d'interconnexion, etc, on ne peut parler de parallélisme sans évoquer les problèmes liés aux algorithmes parallèles et aux langages d'expression du parallélisme. Il importe en effet, pour un problème donné, de lui trouver une solution algorithmique utilisant au maximum les possibilités offertes par la machine parallèle cible choisie et de pouvoir l'exprimer dans un langage de programmation bien approprié.

C'est un problème difficile. Les algorithmes efficaces sur une machine séquentielle ne sont pas nécessairement les plus efficaces sur une machine parallèle. La décomposition de l'algorithme en parties parallélisables doit prendre en compte la manière dont sont réalisés l'accès mémoire et le réseau d'interconnexion afin d'effectuer une "bonne" décomposition et de pouvoir exprimer correctement les éventuelles communications entre les parties parallèles de l'algorithme.

De nombreux algorithmes parallèles ont été publiés principalement dans le domaine numérique (*cf.* [AFC 83]). Il apparait, dans le cadre du parallélisme, que l'on ne peut plus considérer les algorithmes indépendamment des architectures sur lesquelles ils sont réalisés. De plus, les langages d'expression du parallélisme ne sont pas toujours bien adaptés et suffisamment élaborés.

Pour les calculateurs vectoriels, comme le CRAY-1 par exemple, les propositions de langage sont basées sur un langage de haut-niveau (en général FORTRAN) auquel sont ajoutées des opérations de manipulation de vecteurs ou tableaux (par exemple : produit scalaire de deux vecteurs, détermination de l'élément maximum d'un vecteur, etc). Ces opérations peuvent être utilisées explicitement par le programmeur ou directement par le compilateur effectuant une vectorisation.

La vectorisation consiste à déterminer le parallélisme implicite d'un programme séquentiel et à le transformer pour produire un code définissant les opérations vectorielles réalisables (*cf.* [MEY 80], [BOM 81], [BOS 82]). La vectorisation n'est pas chose facile principalement au niveau des itérations et des branchements. Elle peut être réalisée directement par un compilateur qui prend en entrée un programme séquentiel et restitue un code objet vectorisé. En raison des difficultés liées à la vectorisation, ce code n'est pas toujours optimal. L'autre alternative est que le programmeur effectue cette vectorisation "à la main" (*cf.* [BOS 81-a] et [BOS 81-b]).

Au niveau des multiprocesseurs, outre les langages comme LAU développés pour les machines cadencées par les données (*cf.* [BER-al 82]), les principaux langages proposés, comme CSP (*cf.* [HOA 78]) et ADA (*cf.* [ICH-al 79]), permettent de décrire les différents processus indépendamment ainsi que leurs relations. L'outil offert par ces langages est un outil de synchronisation basé sur le rendez-vous, comme les opérations d'envoi et de réception de données proposées par CSP.

Une autre approche consiste à définir les relations entre processus, non plus en terme de synchronisation, mais par l'expression de communications. (*cf.* [JUL 83]).

CHAPITRE 2

LES ARCHITECTURES SYSTOLIQUES

I. Présentation

Une architecture (ou réseau) systolique est une structure parallèle constituée d'un ensemble de processeurs élémentaires, ou cellules, interconnectés régulièrement et localement; chaque cellule exécute une opération simple. Le nombre d'opérations différentes est limité : plusieurs cellules, éventuellement toutes, sont identiques. Les flots de données circulent entre les cellules de façon régulière, éventuellement à des vitesses et dans des directions différentes. La communication avec l'extérieur, c'est-à-dire avec l'ordinateur "hôte" auquel est relié le réseau, ne se fait qu'aux "cellules des extrémités". Par exemple, sur la figure 2.1(a), seules les cellules 1 et n sont des ports d'entrée-sortie pour le système. Le fonctionnement d'un système systolique est synchrone : un ensemble de calculs élémentaires est effectué sur des cellules différentes au même instant.

Un réseau systolique peut être unidimensionnel ou bidimensionnel ainsi que l'illustre la figure 2.1.

La circulation pipelinée des données dans le réseau implique qu'une donnée, une fois acquise par une cellule d'entrée du réseau, est utilisée dans toutes les cellules où elle transite. Sa valeur peut éventuellement être modifiée par le calcul réalisé dans la cellule. Les seuls problèmes intéressants à résoudre par un système systolique sont donc ceux pour lesquels une donnée est utilisée dans plusieurs opérations élémentaires. Ce sont les calculs tributaires de la vitesse de calcul (compute-bound computations dans [KUN 82]) où le nombre total d'opérations est plus grand que le nombre total d'éléments d'entrée-sortie. Pour cette classe de problèmes, l'approche systolique permet de limiter les entrées-sorties avec le calculateur hôte, coûteuses en temps. Toute donnée à usage multiple est acquise une seule fois puis pipelinée à travers les cellules.

Nous distinguons (cf. [KUN 82]), deux types de réseaux systoliques :

— ***Les réseaux semi-systoliques***, caractérisés par des communications globales de données soit avec diffusion, soit avec collecteur (figure 2.2). Dans les réseaux avec diffusion, une donnée est envoyée à plusieurs cellules qui peuvent alors l'utiliser simultanément. Dans les réseaux avec collecteur, le résultat est calculé à partir de ses résultats partiels calculés simultanément dans des cellules différentes.

— ***Les réseaux systoliques (purs)*** sont caractérisés par l'absence de communications globales. Chaque ressource (donnée ou résultat) circule de cellule en cellule ou est locale à une cellule.

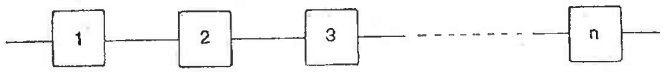
REMARQUE. — Nous ne nous intéressons pas dans l'étude qui suit aux réseaux semi-systoliques avec collecteur.

Les réseaux semi-systoliques avec diffusion présentent l'inconvénient de n'être pas facilement extensibles en raison de la présence de bus pour les communications globales de données, ce qui crée des problèmes de temps d'accès aux données par les différentes cellules (ralentissement de l'horloge du système, risque d'asynchronisme).

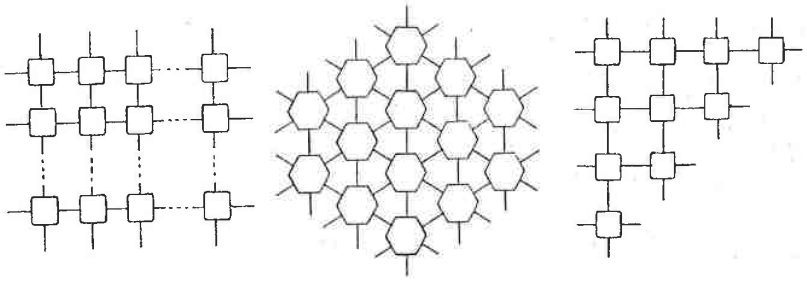
Au contraire, dans les réseaux systoliques purs, les communications locales simplifient les problèmes de synchronisation et de contrôle. Lorsque le résultat est local à une cellule, il est calculé par accumulations successives sur cette cellule. Il faut donc ajouter un chemin de sortie systolique supplémentaire pour extraire chaque résultat de sa cellule après calcul.

De manière classique, on peut dire qu'un algorithme est la description de la composition d'opérations de base. Sa mise en œuvre nécessite un interprète qui utilise cette description pour enchainer les calculs élémentaires pour une valeur des données. Sur une machine de type Von Neumann classique pour laquelle l'architecture est simple (une unité centrale, une unité de contrôle), l'interprète associé à l'algorithme est important. Par contre, sur une machine systolique, l'architecture est beaucoup plus complexe (nombreux opérateurs parallèles interconnectés) mais l'interprète est très simple, réduit aux signaux de synchronisation permettant le fonctionnement simultané correct des différents opérateurs.

La démarche de conception d'architectures systoliques se situe à l'opposé de celle de programmation classique. Habituellement, on utilise une architecture existante "figée" sur laquelle on peut réaliser de nombreux algorithmes par l'intermédiaire d'une structure de contrôle complexe. Dans l'approche systolique, on cherche au contraire à réaliser une architecture



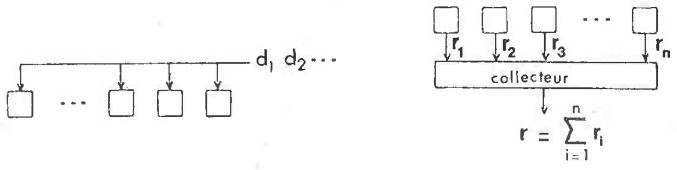
(a) Réseau unidimensionnel ou linéaire



rectangulaire hexagonal triangulaire

(b) Réseaux bidimensionnels

FIGURE 2.1. Réseaux systoliques



avec diffusion avec collecteur

FIGURE 2.2. Exemples de réseaux semi-systoliques

spécialisée adaptée à un seul algorithme. Dans ce cas, la structure de contrôle est quasi inexistante.

Ce type d'approche n'a de sens qu'en raison :
— des progrès technologiques au niveau de circuits intégrés qui permettent de réaliser efficacement ces architectures.
— des contraintes imposées aux structures systoliques qui sont composées nombreux processeurs identiques connectés localement, ce qui limite l'effort de conception des circuits grâce à leur reproduction en nombreux exemplaires.

Notons, qu'en raison de la structure régulière des architectures systoliques, tout problème ne peut s'exprimer par un algorithme systolique. Dans la suite, nous appelons *problème systolisable* un problème pour lequel il existe un algorithme systolique.
Nous allons présenter :
— un ensemble de réseaux systoliques linéaires solution du produit de convolution.
— un ensemble de réseaux bidimensionnels pour le produit matriciel.

II. Un exemple de réseaux linéaires : le produit de convolution

Le calcul du produit de convolution est défini par : étant donné la suite de réels $(x_i)_{i=0,\ldots,n}$ et la suite de coefficients réels $(w_k)_{k=0,\ldots,m}$, calculer la suite $(y_i)_{i=0,\ldots,n-m}, n > m$, avec

$$y_i = \sum_{k=0}^{m} w_k \times x_{i+k}$$

On prend, pour les exemples donnés ci-dessous, $n = 7$ et $m = 2$.
Le problème peut s'exprimer par la relation de récurrence :

$$\begin{cases} y_i^k = y_i^{k-1} + w_l \times x_{i+l} & i = 0, \ldots, n-m \quad k = 0, \ldots, m \\ y_i^{-1} = 0 \end{cases}$$

avec $l = k$ ou $l = m - k$.

Le choix $l = k$ correspond au calcul dans l'ordre croissant de l'indice k :

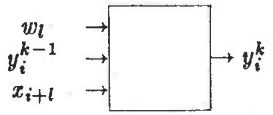
$$y_i = w_0 x_i + w_1 x_{i+1} + \cdots + w_m x_{i+m}$$

tandis que le choix $l = m - k$ correspond au calcul dans l'ordre décroissant de l'indice k :

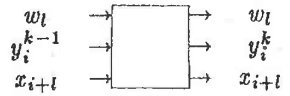
$$y_i = w_m x_{i+m} + w_{m-1} x_{i+m-1} + \cdots + w_0 x_i$$

En raison de la commutativité de l'addition, tout autre choix dans l'ordre des calculs est possible. Mais les deux seuls intéressants sont ceux retenus ci-dessus. En effet, les architectures systoliques intéressantes sont celles où les données d'une suite, qui transitent sur un flot, circulent dans un ordre régulier des indices, soit croissant, soit décroissant, afin de faciliter la lecture en mémoire de ces données, généralement stockées en des emplacements contigus. Les exemples présentés dans ce paragraphe correspondent à $l = k$.

La conception d'une architecture systolique associée à un problème consiste à réaliser simultanément certaines des opérations élémentaires définies par la récurrence ci-dessus en utilisant des opérateurs (ou cellules). On distingue deux catégories de cellules :
— les cellules fonctionnelles de la forme :

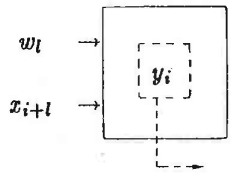


Elles réalisent la fonction de récurrence en calculant la valeur de sortie y_i^k à partir des valeurs d'entrées y_i^{k-1}, w_l, x_{i+l}. Une cellule de cette catégorie peut éventuellement être munie d'une zone de mémoire locale contenant une constante, ici w_l (cf. exemple 2) ou x_{i+l}. Si une donnée w_l ou x_{i+l} doit être réutilisée plus tard par un autre opérateur, on ajoute à la cellule une sortie restituant la valeur de la donnée d'entrée :



La valeur initiale d'une donnée y_i sur le flot, avant d'entrer dans le réseau, correspond à l'initialisation de la récurrence, c'est-à-dire ici 0.

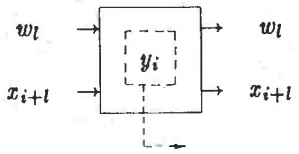
— les cellules à mémoire de la forme :



La fonction de récurrence à calculer est appliquée sur la zone de mémoire locale utilisée comme variable : son contenu est modifié à chaque activation de la cellule (cf. exemple 1). Pour ces cellules, chaque résutat élémentaire y_i à calculer reste dans leur mémoire locale (représentée en pointillé sur le dessin ci-dessus). Lorsque la cellule a été k fois active, le contenu de sa zone mémoire correspond au k-ième pas de la récurrence, y_i^k.

La valeur initiale de la zone mémoire d'une cellule correspond à l'initialisation de la récurrence, ici la valeur 0.

Quand tous les calculs sont terminés, les résultats $y_i^m, i = 0, \ldots, n-m$ doivent être extraits du réseau en utilisant le chemin de sortie supplémentaire visualisé en pointillé sur le dessin ci-dessus. Les données peuvent être utilisées dans plusieurs cellules du réseau. Dans ce cas, elles transitent sur un flot :



Selon la nature de la relation de récurrence, le réseau peut avoir plusieurs types d'opérateurs ou cellules. Pour le produit de convolution décrit ci-dessus, tous les calculs élémentaires sont les mêmes. Les différents réseaux systoliques associés sont donc constitués d'un seul type de cellules de calcul, différent selon les réseaux.

Pour une relation de récurrence élémentaire donnée, $y_i^k = y_i^{k-1} + w_l \times x_{i+l}$, i fixé, les pas successifs de la récurrence sont effectués à des instants successifs. Par contre, l'utilisation du parallélisme, réalisé par la présence de plusieurs opérateurs ou cellules, doit permettre d'effectuer un pas de

plusieurs récurrences élémentaires différentes au même instant, sur des opérateurs différents.

Déterminer un réseau systolique consiste à composer un ensemble d'opérateurs $O_0, \ldots, O_q$ en associant pour chaque instant t donné, à un opérateur $O_p(p = 0, \ldots, q)$, un terme y_i^{k-1}, un terme w_l et un terme x_{i+l} où $l = k$ ou $l = m - k$, q est une constante dépendant de n et m, p et t sont définis en fonction de i et k.

Considérons les exemples de composition suivants :

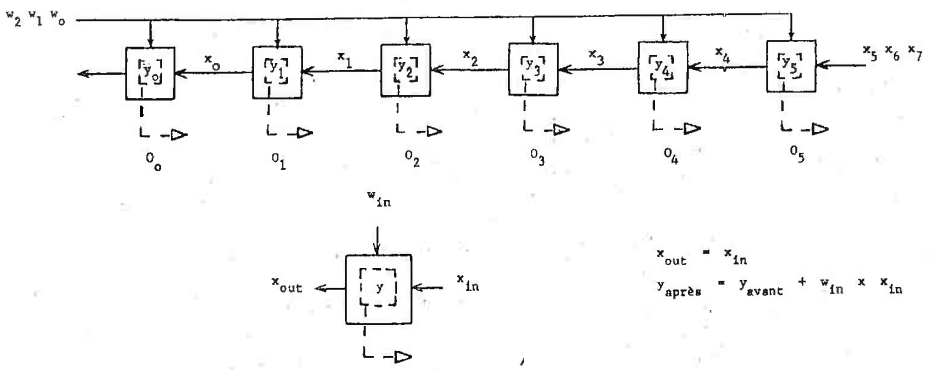
EXEMPLE 1. — Prenons $l = k, q = n - m, p = i$ et $t = k$. On associe alors, pour un instant $t = k$ donné, à un opérateur O_i, les termes y_i^{k-1}, w_k et x_{i+k} avec $i = 0, \ldots, n - m$ et $k = 0, \ldots, m$.

Le réseau correspondant est constitué de $q + 1 = n - m + 1$ cellules. A un instant $t = k$ donné, la cellule O_i utilise le terme y_i^{k-1} pour calculer y_i^k. Les cellules étant actives simultanément, les k-ième pas de toutes les récurrences élémentaires sont calculés en même temps sur les différentes cellules. Une cellule O_i utilise successivement y_i^{-1} à $t = 0$, y_i^0 à $t = 1$, y_i^1 à $t = 2$,etc. Elle calcule donc les pas successifs d'une même récurrence élémentaire. Le réseau est alors constitué de cellules à mémoire : le contenu de la mémoire locale de O_i à l'instant k est le résultat des k premières étapes de la récurrence $y_i^k = y_i^{k-1} + w_k \times x_{i+k}$.

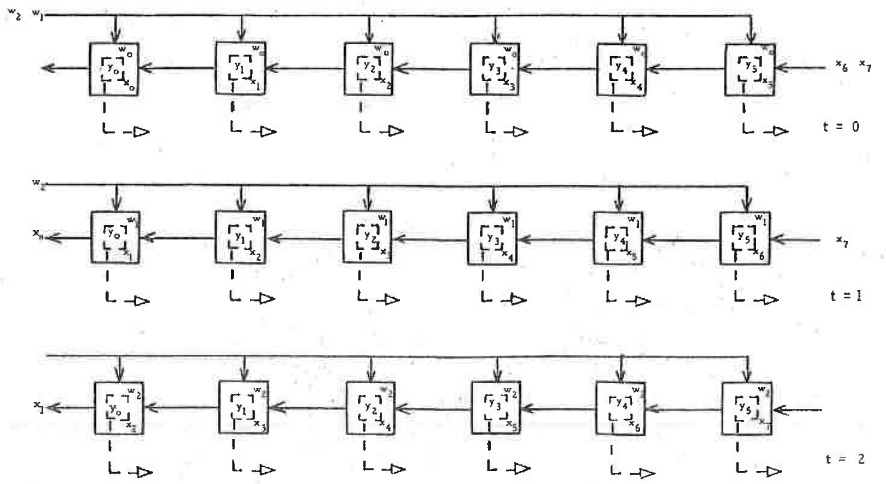
De plus à l'instant k, toutes les cellules O_i, $i = 0, \ldots, n - m$ utilisent le terme w_k. Ceci se traduit dans le réseau par la diffusion des données $w_k, k = 0, \ldots, m$.

Prenons $i' = i-1, k' = k+1$. On a alors $x_{i+k} = x_{i'+k'}$. Or la donnée x_{i+k} est utilisée par la cellule O_i à $t = k$, tandis que la donnée $x_{i'+k'} = x_{i+k}$ est utilisée par la cellule $O_{i'} = O_{i-1}$ à $t' = k' = k + 1$. Les données $(x_i)_{i=0,\ldots,n}$ circulent donc dans le réseau d'une cellule O_p vers une cellule $O_{p-1}, p = 1, \ldots, q$.

Le réseau ainsi décrit est celui de la figure 2.3(a). Les données w_{in} et x_{in} correspondent aux valeurs de w et x, présentes sur le chemin de communication avant l'exécution de l'opération sur la cellule, tandis que x_{out} correspond à la valeur de x présente sur le chemin de communication issu de la cellule après l'exécution de l'opération. De même y_{avant} (resp y_{apres}) représente la valeur de la zone mémoire de la cellule avant (resp. après) l'exécution de l'opération. L'égalité $x_{out} = x_{in}$ traduit le fait que les données x ne sont pas modifiées par leur passage dans une cellule. Les cellules



(a) Réseau systolique



(b) Simulation d'exécution

FIGURE 2.3. Réseau systolique associé au produit de convolution

étant à mémoire, un chemin de sortie, en pointillé, est nécessaire pour extraire après exécution le contenu de la mémoire locale à chaque cellule O_i. Ce contenu est ici le résultat élémentaire y_i^m calculé par récurrence.

La simulation de l'exécution de ce réseau à la figure 2.3(b) permet de décrire son fonctionnement et la façon dont circulent les données. Le temps d'exécution du réseau est de $m+1$ unités de temps puisque $t = k$ et $k = 0, \ldots, m$. Il faut y ajouter le temps d'extraction des résultats y_i, tous obtenus à $t = m$. Les données $(x_i)_{i=0,\ldots,n}$ n'auront terminé leur transit dans le réseau qu'à $t = m + q + 1 = n + 1$. La situation présentée figure 2.3 nécessite bien sûr un temps d'amorçage permettant le transit des (x_i) jusqu'à leur position initiale décrite à la figure 2.3(a).

EXEMPLE 2. — Prenons $l = k, q = m, p = k$ et $t = i + 2k$. On associe alors, pour un instant $t = i + 2k$, à un opérateur O_k, les termes y_i^{k-1}, w_k et x_{i+k} avec $i = 0, \ldots, n - m$ et $k = 0, \ldots, m$.

A tout instant, l'opérateur O_k utilise le terme w_k. Cela signifie que la donnée w_k est constante de la cellule O_k. Le réseau est donc constitué de cellules fonctionnelles contenant une constante locale w_k.

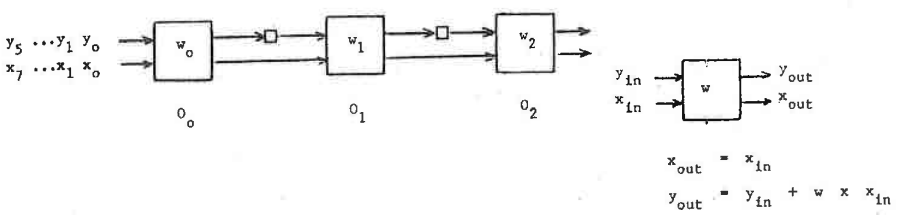
Pour la suite de données $(x_i)_{i=0,\ldots,n}$, on a les résultats suivants :

— la donnée x_{i+k} est utilisée par la cellule O_k à l'instant $t = i+2k$. Notons $i' = i - 1$ et $k' = k + 1$. La donnée $x_{i+k} = x_{i'+k'}$ est utilisée par la cellule $O_{k'} = O_{k+1}$ à l'instant $t' = i' + 2k' = i + 2k + 1 = t + 1$. La suite de données $(x_i)_{i=0,\ldots,n}$ transite donc dans le réseau d'une cellule O_p vers une cellule $O_{p+1}, p = 0, \ldots, q - 1$.

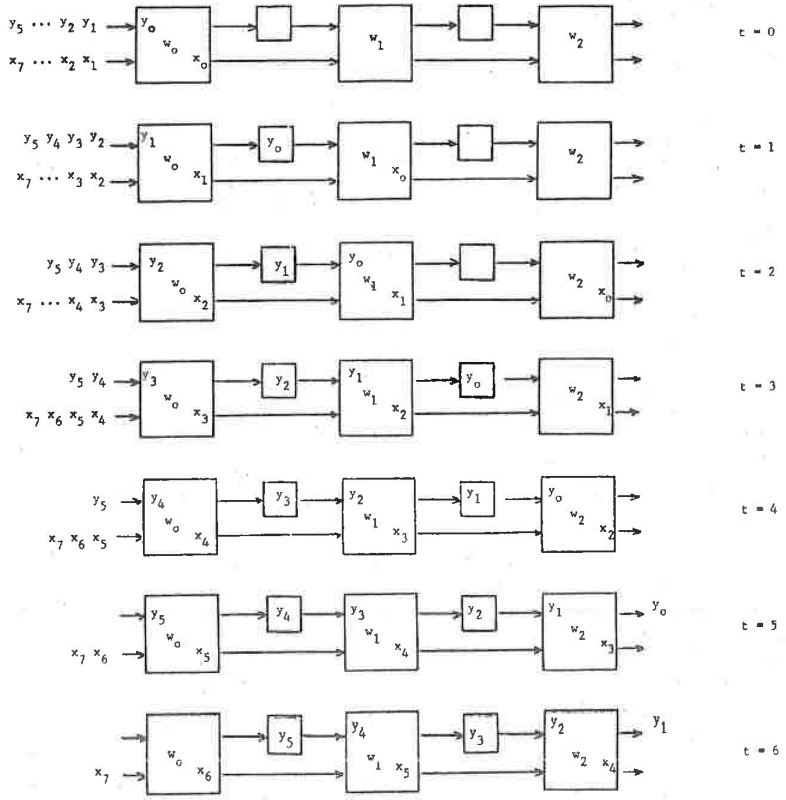
— à l'instant $t = i + 2k$, la cellule O_k utilise la donnée x_{i+k} tandis qu'à l'instant $t' = t + 1 = i' + 2k'$ avec $i' = i + 1$ et $k' = k$, la cellule $O_{k'} = O_k$ utilise la donnée $x_{i'+k'} = x_{i+k+1}$. Cela signifie que les données $(x_i)_{i=0,\ldots,n}$ sont ordonnées dans l'ordre croissant de leur indice sur leur flot comme le montre la figure 2.4.

Pour la suite de résultats $(y_i)_i = 0, \ldots, n - m$, les résultats sont les suivants :

— la donnée y_i^{k-1} est utilisée par la cellule O_k à l'instant $t = i + 2k$ pour calculer le résultat partiel $y_i^k = y_i^{(k+1)-1}$ qui lui sera utilisé par la cellule O_{k+1} à l'instant $t' = i + 2(k + 1) = i + 2k + 2 = t + 2$. Cela signifie que le flot des $(y_i)_{i=0,\ldots,n-m}$ transite dans le réseau d'une cellule O_p vers une cellule O_{p+1} mais qu'un résultat partiel calculé sur O_p à un instant t n'est utilisé par la cellule adjacente O_{p+1} qu'à l'instant $t + 2$. Cette dernière condition est traduite par la présence de cellules de retard (petite cellule sur le flot entre deux cellules consécutives sur la figure 2.4). Elle symbolise le fait que le flot considéré avance deux fois plus



(a) Réseau systolique



(b) Début de la simulation d'exécution

FIGURE 2.4. Réseau systolique associé au produit de convolution

lentement que celui des $(x_i)_{i=0,\ldots,n}$. Aucun calcul n'est effectué dans les cellules retard qui restituent en sortie la valeur lue à l'entrée.

— à l'instant $t = i + 2k$, la cellule O_k utilise le résultat partiel y_i^{k-1} tandis qu'à l'instant $t' = t + 1 = i' + 2k'$ avec $i' = i + 1$ et $k' = k$, la cellule $O_{k'} = O_k$ utilise le résultat $y_{i'}^{k'-1} = y_{i+1}^{k-1}$. Les données $(y_i)_{i=0,\ldots,n-m}$ sont donc ordonnées dans l'ordre croissant de leurs indices sur leur flot.

Le réseau ainsi décrit est représenté à la figure 2.4. Après un "temps d'amorçage" de $2q = 2m$ unités de temps, toutes les cellules du réseau sont actives à chaque pas et un résultat élémentaire $y_i, i = 0, \ldots, n - m$ sort du réseau à chaque intervalle de temps. Le temps d'exécution total du réseau correspond à : temps d'amorçage (2m) + nombre de résultats (n-m+1). Il est donc de $n + m + 1$ unités de temps.

EXEMPLE 3. — Prenons $l = k, q = n, p = k-i+n-m$ et $t = i+k$. On associe alors, pour un instant $t = i + k$ donné, à un opérateur $O_{k-i+n-m}$, les termes y_i^{k-1}, w_k et x_{i+k} avec $i = 0, \ldots, n - m$ et $k = 0, \ldots, m$.

Toute donnée x_{i+k} est utilisée à l'instant $t = i + k$ et cela par plusieurs cellules. Il y a donc diffusion des données de la suite $(x_i)_{i=0,\ldots,n}$ dans l'ordre croissant des indices.

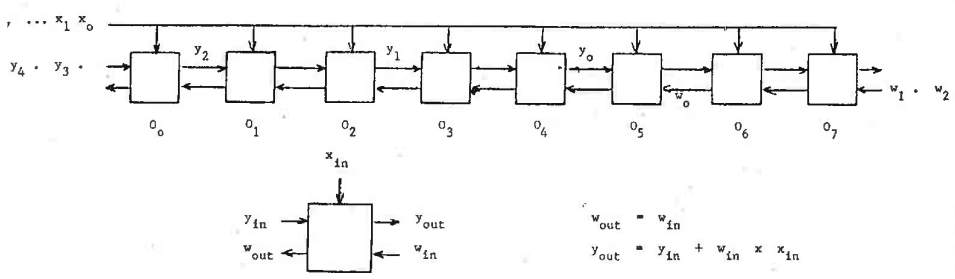
Pour la suite de données $(w_k)_{k=0,\ldots,n}$, on a les résultats suivants :

— la donnée w_k est utilisée par la cellule $O_{k-i+n-m}$ à l'instant $t = i + k$. Prenons $i' = i+1$ et $k' = k$, la donnée $w_{k'} = w_k$ est utilisée par la cellule $O_{k'-i'+n-m} = O_{k-i+n-m-1}$ à l'instant $t' = i'+k' = i+k+1 = t+1$. La suite de donnée $(w_k)_{k=0,\ldots,m}$ transite donc dans le réseau d'une cellule O_p vers une cellule $O_{p-1}, p = 1, \ldots, q$ comme le montre la figure 2.5.

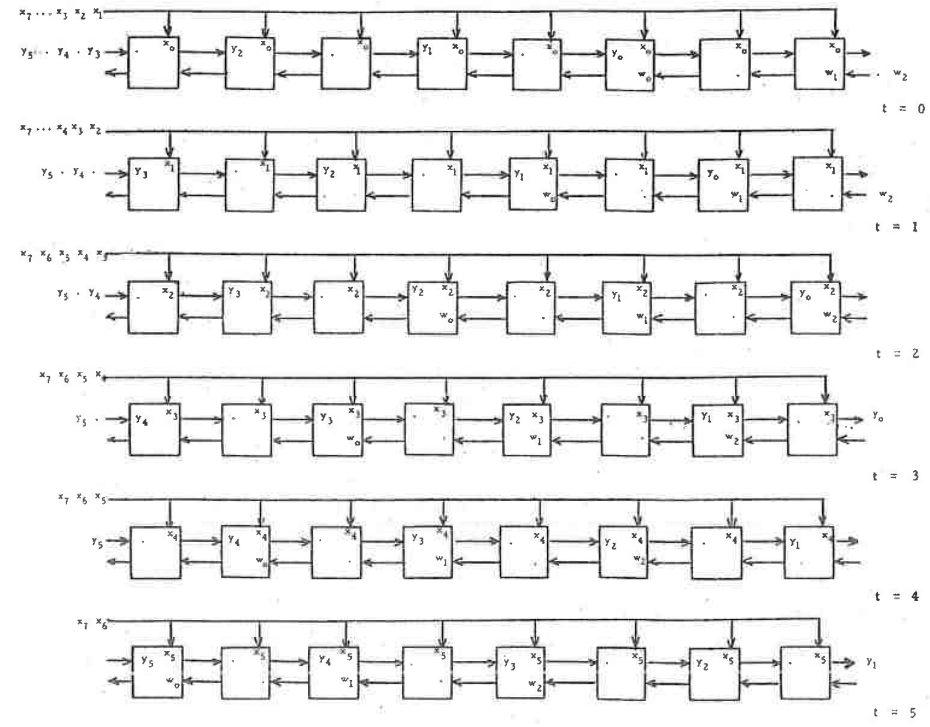
— à l'instant $t = i + k$ la cellule $O_{k-i+n-m}$ utilise la donnée w_k tandis qu'à $t' = t + 2 = i' + k'$ avec $i' = i + 1$ et $k' = k + 1$, la cellule $O_{k'-i'+n-m} = O_{k-i+n-m}$ utilise la donnée $w_{k'} = w_{k+1}$. Cela signifie que les données $(w_k)_{k=0,\ldots,m}$ transitent sur leur flot dans l'ordre croissant de leurs indices mais avec deux intervalles de temps entre l'acquisition de deux w_k successifs car une cellule donnée utilise un w_k à l'instant t et w_{k+1} à l'instant $t + 2$. Cela est traduit par la présence du point (.) entre deux w_k consécutifs sur le flot (cf. figure 2.5).

De même pour la suite des $(y_i)_{i=0,\ldots,n-m}$, on constate que :

— la donnée y_i^{k-1} est utilisée par la cellule $O_{k-i+n-m}$ à l'instant $t = i + k$ pour calculer le résultat partiel y_i^k qui est lui-même utilisé par la cellule



(a) Réseau systolique



(b) Début de la simulation d'exécution

FIGURE 2.5. Réseau systolique pour le produit de convolution

$O_{k+1-i+n-m} = O_{k-i+n-m+1}$ à l'instant $t' = i + k + 1 = t + 1$. La suite $(y_i)_{i=0,\ldots,n-m}$ transite donc d'une cellule O_p vers une cellule O_{p+1}, $p = 0, \ldots, q-1$.

— à l'instant $t = i + k$, la cellule $O_{k-i+n-m}$ utilise la donnée y_i^{k-1} tandis qu'à l'instant $t' = t + 2 = i' + k'$ avec $i' = i + 1$ et $k' = k + 1$, cette même cellule $O_{k'-i'-n-m} = O_{k-i+n-m}$ utilise la donnée $y_{i'}^{k'-1} = y_{i+1}^{k}$. Cela signifie que la suite $(y_i)_{i=0,\ldots,n-m}$ transite dans le sens croissant de ces indices avec un espacement (.) entre deux y_i consécutifs.

Le réseau correspondant est illustré à la figure 2.5. Il est constitué de cellules fonctionnelles sans constante locale. En raison de l'espacement sur le flot des $(y_i)_{i=0,\ldots,n-m}$, après le temps d'amorçage de $q = n$ unités de temps, les résultats élémentaires consécutifs sortent du réseau toutes les deux unités de temps. Le temps total d'exécution du réseau correspond à : temps d'amorçage (n) + 2 × nombre de résultats y_i à calculer $(n - m + 1)$ - 1, soit $3n - 2m + 1$ unités de temps.

REMARQUE. — Sur les figures 2.3, 2.4 et 2.5, nous ne faisons pas apparaitre l'indice de récurrence k sur les données $(y_i)_{i=0,\ldots,n-m}$ pour ne pas surcharger le dessin. Il est clair que, lorsque y_{in} d'une cellule correspond au résultat partiel y_i^{k-1}, alors y_{out} correspond au résultat y_i^k qui est utilisé comme entrée (y_{in}) par la cellule suivante. Aussi la valeur correspondant à un y_i donné aux instants successifs de la simulation est-elle à chaque fois différente.

III. Un exemple de réseaux bidimensionnels : le produit matriciel

Le produit matriciel $C = A \times B$ avec A de dimension (m, n), B de dimension (n, p) et C de dimension (m, p) est défini par

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj} \qquad i = 1, \ldots, m \quad j = 1, \ldots, p$$

Comme pour le produit de convolution, le problème peut s'exprimer par une récurrence :

$$\begin{cases} c_{ij}^{k} = c_{ij}^{k-1} + a_{il} \times b_{lj}, & k = 1, \ldots, n \quad i = 1, \ldots, m \quad j = 1, \ldots, p \\ c_{ij}^{0} = 0 \end{cases}$$

avec $l = k$ ou $l = n - k + 1$.

Comme pour l'exemple du paragraphe II, le choix $l = k$ correspond à l'exécution de la récurrence dans l'ordre croissant de l'indice k, tandis que le choix $l = n - k + 1$ correspond à son exécution dans l'ordre décroissant de l'indice k.

Dans les exemples présentés ci-dessous, on prend $m = 2, n = 2$ et $p = 3$.

Les réseaux systoliques associés au problème sont bidimensionnels car les résultats élémentaires c_{ij} à calculer sont définis par deux indices. Un opérateur ou cellule est donc caractérisé par deux indices $O_{r,s}$ correspondant à la position de l'opérateur dans un plan.

Les calculs élémentaires étant tous de la même forme, un réseau quelconque associé au problème est constitué d'un seul type de cellules, qui peuvent être soit fonctionnelles, soit à mémoire.

Déterminer un réseau consiste à composer un ensemble d'opérateurs $O_{r,s}$ avec $r \in \{1, \ldots, q_1\}$ et $s \in \{1, \ldots, q_2\}$ en associant pour chaque instant t donné, à un opérateur $O_{r,s}$, un terme c_{ij}^{k-1}, un terme a_{il} et un terme b_{lj} avec $l = k$ ou $l = n - k + 1$, q_1 et q_2 des constantes dépendant de n, m, p et r, s, t définis en fonction de i, j, k.

EXEMPLE 1. — Prenons $l = k, q_1 = m, q_2 = p, r = i, s = j$ et $t = k$. On associe, pour un instant $t = k$ donné, à une cellule $O_{i,j}$, les termes c_{ij}^{k-1}, a_{ik} et b_{kj}.

On constate qu'une cellule $O_{i,j}$, i et j fixés, utilise successivement c_{ij}^{0} à $t = 1$, c_{ij}^{1} à $t = 2$, c_{ij}^{2} à $t = 3$,etc. Le réseau est donc constitué de cellules à mémoire dont le contenu de la zone de mémoire locale de $O_{i,j}$ correspond à la valeur c_{ij} aux différents pas de la récurrence.

A un instant $t = k$ donné, un terme a_{ik}, i fixé, est utilisé simultanément par toutes les cellules de la forme $O_{i,j}, j \in \{1, \ldots, q_2\}$. Il y a donc diffusion des données a_{ik}. Il en est de même pour les données b_{kj} utilisées simultanément par toutes les cellules de la forme $O_{i,j}, i \in \{1, \ldots, q_1\}$.

Le réseau correspondant est donc un réseau semi-systolique avec double diffusion à la fois des données a_{ik} et des données b_{kj}. Il est représenté à la figure 2.6.

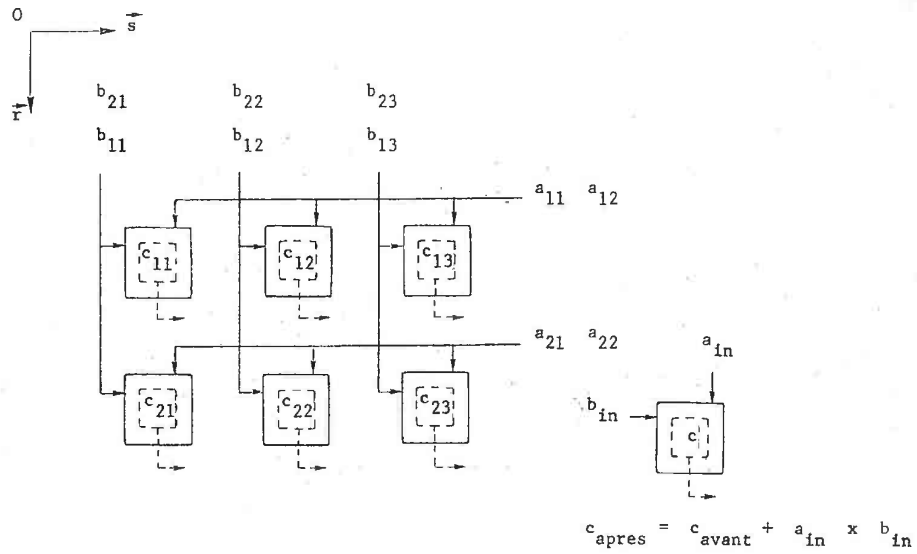


FIGURE 2.6. Réseau semi-systolique avec double diffusion

Tous les réseaux de ce paragraphe sont représentés dans un plan muni d'un repère $(O, \vec{r}, \vec{s})$, $(O, \vec{r})$ étant un axe vertical orienté vers le bas et $(O, \vec{s})$ un axe horizontal orienté vers la droite comme le montre la figure 2.6.

D'autres choix pour r, s et t conduisent à de nouveaux réseaux systoliques qui peuvent être semi-systoliques avec une diffusion (cf.figures 2.7 et 2.8) ou systoliques purs (cf.figures 2.9 ou 2.10). Leurs cellules peuvent être fonctionnelles (cf. figures 2.8, 2.9, 2.10) ou à mémoire (cf.figure 2.7).

Nous ne détaillons pas ici comment sont obtenus et comment fonctionnent ces réseaux, préférant illustrer cela en détail sur le réseau classique, appelé réseau hexagonal, proposé par de nombreux auteurs (cf. [KUN 80], [MEC 80] et [QUI 83]).

EXEMPLE 2. — Obtention du réseau hexagonal (cf. figure 2.11).

Prenons $l = k, q_1 = n + p - 1, q_2 = m + n - 1, r = j - k + n, s = k - i + m$ et $t = i + j + k$. Pour un instant $t = i + j + k$ donné, on associe, à une cellule $O_{r,s} = O_{j-k+n,k-i+m}$, les termes c_{ij}^{k-1}, a_{ik} et b_{kj} avec $i = 1, \ldots, m, j = 1, \ldots, p, k = 1, \ldots, n$.

Examinons successivement les suites de données.

Pour la suite $(c_{ij})_{i=1,\ldots,m,j=1,\ldots,p}$ on a :

— la donnée c_{ij}^{k-1} est utilisée sur la cellule $O_{r,s} = 0_{j-k+n,k-i+m}$ à l'instant $t = i+j+k$ pour calculer la valeur de c_{ij}^{k}, qui est elle-même utilisée sur la cellule $O_{j-(k+1)+n,(k+1)-i+m} = 0_{r-1,s+1}$ à l'instant $t' = i+j+(k+1) = t+1$. Les données c_{ij} transitent donc en diagonale dans le réseau d'une cellule $O_{r,s}$ vers une cellule $0_{r-1,s+1}$.

— à l'instant $t = i+j+k$, la cellule $O_{r,s} = O_{j-k+n,k-i+m}$ utilise la donnée c_{ij}^{k-1} tandis qu'à l'instant $t' = t+3 = i'+j'+k'$ (avec $i' = i+1, j' = j+1, k' = k+1$), la cellule $O_{j'-k'+n,k'-i'+m} = O_{r,s}$ utilise la donnée $c_{i'j}^{k'-1} = c_{i+1,j+1}^{k}$.

Le réseau étant bidimensionnel, la circulation des données d'une suite est réalisée par un ensemble de flots parallèles permettant de desservir toutes les cellules du réseau. Chaque donnée d'une suite appartient à un seul de ces flots. Pour la suite (c_{ij}) considérée, une même cellule utilise les données c_{ij}^{k-1} à un instant t et $c_{i+1,j+1}^{k}$ à l'instant $t+3$. Les flots parallèles sont donc constitués par les diagonales de la matrice C.

De plus, trois unités de temps séparant deux opérations successives sur une même cellule, les données sur les flots parallèles associés à la suite (c_{ij}) sont séparées par deux points (.).

Pour la suite $(a_{ik})_{i=1,\ldots,m,k=1,\ldots,n}$ on a :

— La donnée a_{ik} est utilisée par la cellule $O_{r,s} = O_{j-k+n,k-i+m}$ à l'instant $t = i+j+k$. Prenons $i' = i, j' = j+1, k' = k$, la donnée $a_{i'k'} = a_{ik}$ est aussi utilisée par la cellule $O_{j'-k'+n,k'-i'+m} = O_{j-k+n+1,k-i+m} = O_{r+1,s}$ à l'instant $t' = i'+j'+k' = i+j+k+1 = t+1$. Les données a_{ik} transitent donc dans le réseau d'une cellule $O_{r,s}$ vers une cellule $O_{r+1,s}$

— à l'instant $t = i+j+k$, la cellule $O_{r,s} = O_{j-k+n,k-i+m}$ utilise la donnée a_{ik} tandis qu'à l'instant $t' = t+3 = i'+j'+k'$ (avec $i' = i+1, j' = j+1, k' = k+1$) la cellule $O_{j'-k'+n,k'-i'+m} = O_{r,s}$ utilise la donnée $a_{i'k'} = a_{i+1,k+1}$. Les flots parallèles correspondant à la suite (a_{ik}) est donc constitués des diagonales de la matrice A, deux données successives d'un même flot étant séparées par deux points (.) Ces points symbolisent l'espacement des données sur les flots correspondants.

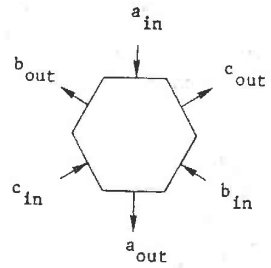
Pour la suite $(b_{kj})_{k=1,\ldots,n,j=1,\ldots,p}$ on a :

— la donnée b_{kj} est utilisée par la cellule $O_{r,s} = O_{j-k+n,k-i+m}$ à l'instant $t = i+j+k$. Prenons $i' = i+1, j' = j, k' = k$ la donnée $b_{k'j'} = b_{kj}$

est aussi utilisée par la cellule $O_{j'-k'+n,k'-i'+m} = O_{j-k+n,k-i+m-1} = O_{r,s-1}$ à l'instant $t' = i'+j'+k' = t+1$. Les données b_{kj} transitent donc dans le réseau d'une cellule $O_{r,s}$ vers une cellule $O_{r,s-1}$.

— de plus, en appliquant les mêmes remarques que pour la suite (a_{ik}), on conclut que les flots parallèles correspondant à la suite (b_{kj}) sont constitués par les diagonales de la matrice B, avec deux points(.) entre les données successives d'un même flot.

Le réseau ainsi obtenu est représenté à la figure 2.11. Les deux cellules des extrémités de coordonnées (1,1) et (4,3) dans le repère $(O, \vec{r}, \vec{s})$ ne sont pas représentées car aucun calcul n'y est effectué. Les cellules carrées de ce réseau pourraient être remplacées par des cellules hexagonales équivalentes de la forme :



avec

$$\begin{cases} a_{out} = a_{in} \\ b_{out} = b_{in} \\ c_{out} = c_{in} + a_{in} \times b_{in} \end{cases}$$

Le réseau serait alors le réseau hexagonal régulier tel qu'il est représenté dans la littérature.

L'exécution de ce réseau est simulé à la figure 2.12. Les flots parallèles associés à la matrice A sont verticaux orientés de haut en bas, ceux associés à la matrice B sont horizontaux orientés de gauche à droite et ceux associés à la matrice C sont obliques orientés du bas-droit vers le haut-gauche. Les valeurs initiales des données c_{ij}^{0} correspondent à l'initialisation de la récurrence $c_{ij}^{k} = c_{ij}^{k-1} + a_{ik} \times b_{kj}$. Elles sont donc nulles.

Une cellule est active lorsque ces trois entrées sont pourvues de valeurs effectives. Les cellules actives sont visualisées sur la figure 2.12 par des contours plus épais. Ainsi sur la figure 2.12(b), la cellule $(2,2)$ est la seule cellule active. Elle effectue le premier pas de la récurrence associée à c_{11}, c'est-à-dire $c_{11}^{1} = c_{11}^{0} + a_{11} \times b_{11}$. Le résultat de l'opération c_{11}^{1} n'est pas indiqué sur le dessin pour éviter de le surcharger. Il apparait par contre sur la figure 2.12 (c) où il sert de donnée à la cellule $(1,3)$ qui calcule le deuxième pas de la récurrence : $c_{11}^{2} = c_{11}^{1} + a_{12} \times b_{21}$. A l'instant $t = 3$, trois

cellules sont actives simultanément calculant respectivement $c_{11}^2, c_{12}^1, c_{21}^1$ comme le montre la figure 2.12 (c).

Le calcul d'un résultat élémentaire nécessite deux calculs partiels correspondant aux deux pas de la récurrence. Ces deux pas sont effectués à des instants consécutifs dans des cellules adjacentes (cf. figure 2.12(b) et 2.12(c) pour le résultat élémentaire c_{11}). Lorsqu'un résultat sort du réseau, il est entièrement calculé. Ainsi à l'instant $t = 4$, le résultat élémentaire c_{11} correspondant au calcul de récurrence c_{11}^2 est disponible à la sortie du réseau.

Notons que lorsqu'une valeur transite dans une cellule non active, sa valeur n'est pas modifiée. C'est pourquoi l'indice de récurrence d'un c_{ij} n'est incrémenté de 1 qu'après passage dans une cellule active. Ainsi à $t = 1$, la valeur c_{11}^0 en entrée de la cellule $(3,1)$ n'est pas modifiée. Ainsi, à $t = 2$, la valeur d'entrée de la cellule adjacente $(2,2)$ sur le flot est donc également c_{11}^0. Cette cellule étant active, la valeur de sortie correspondante est $c_{11}^1 = c_{11}^0 + a_{11} \times b_{11}$ que l'on retrouve en valeur d'entrée sur la cellule $(1,3)$ à $t = 3$ (figure 2.12 (c)).

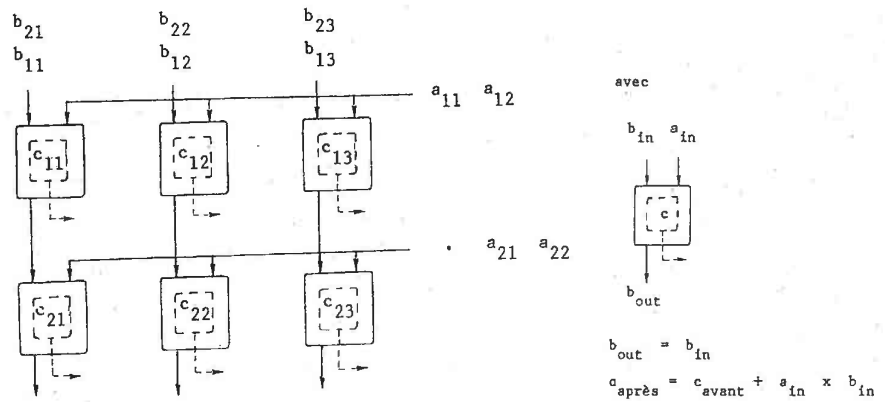


Figure 2.7. Réseau semi-systolique avec une diffusion

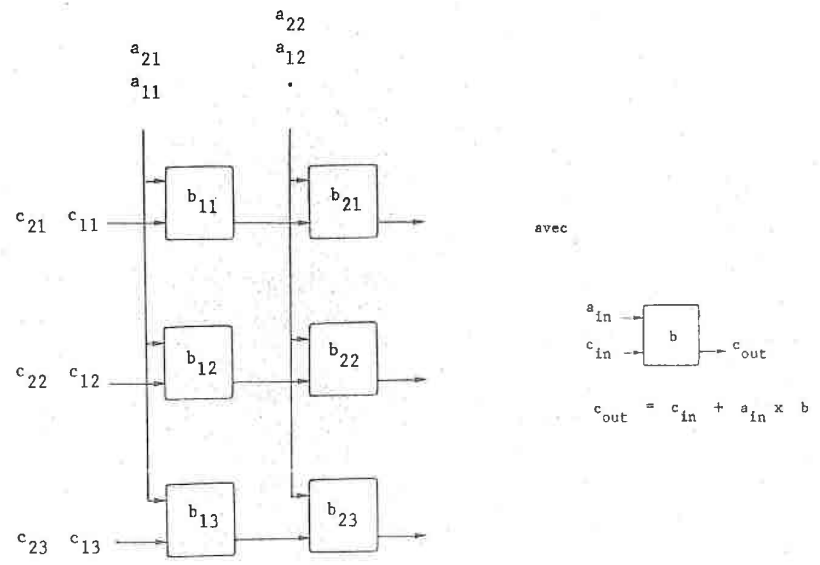


Figure 2.8. Réseau semi-systolique avec une diffusion

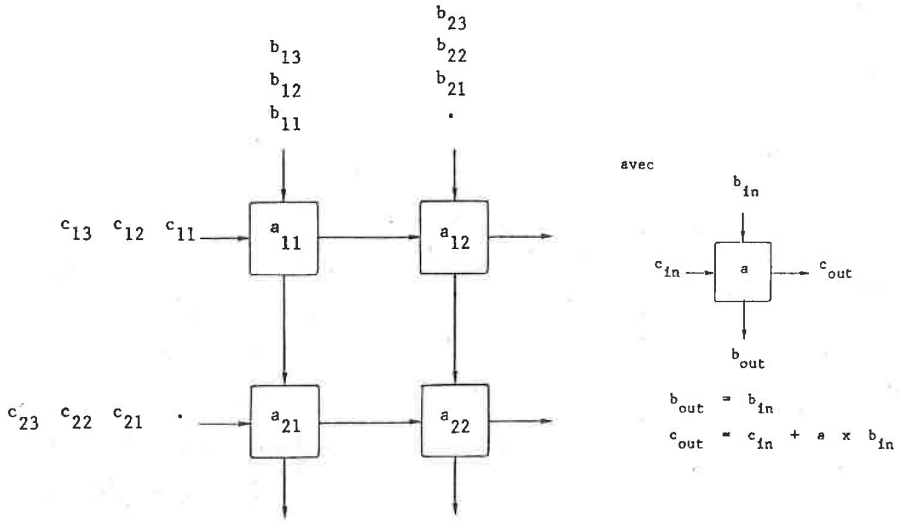


Figure 2.9. Réseau systolique pur

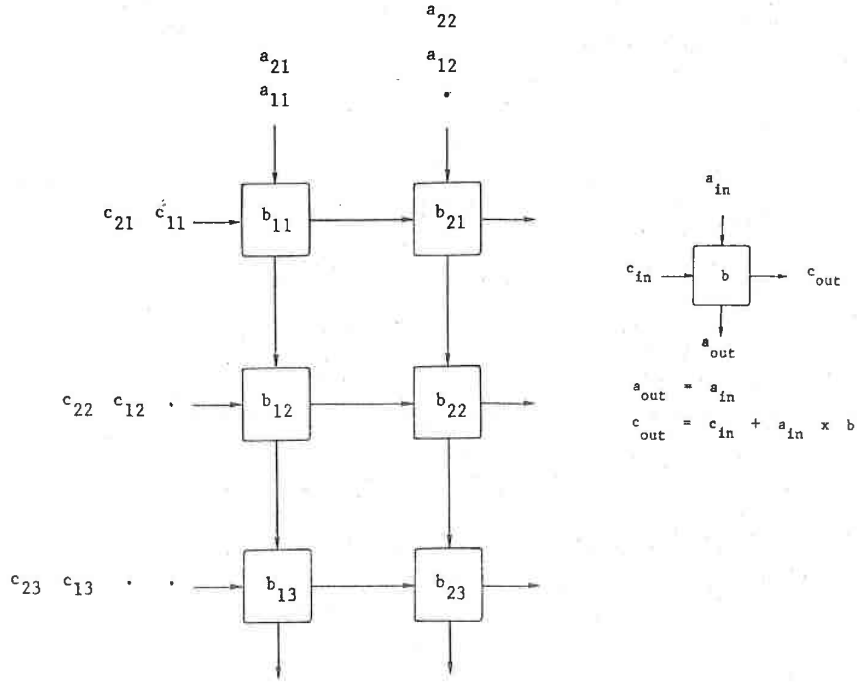


Figure 2.10. Réseau systolique pur

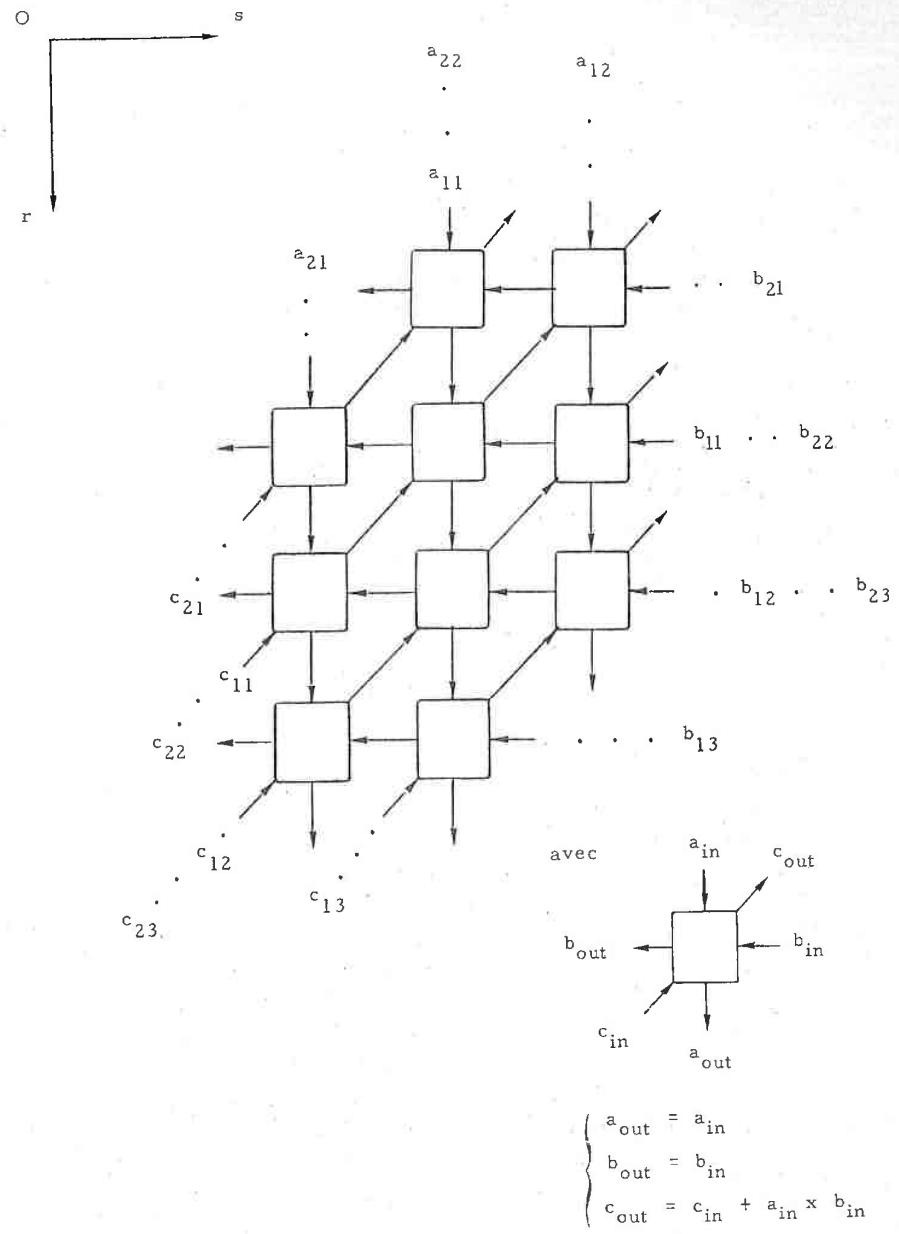
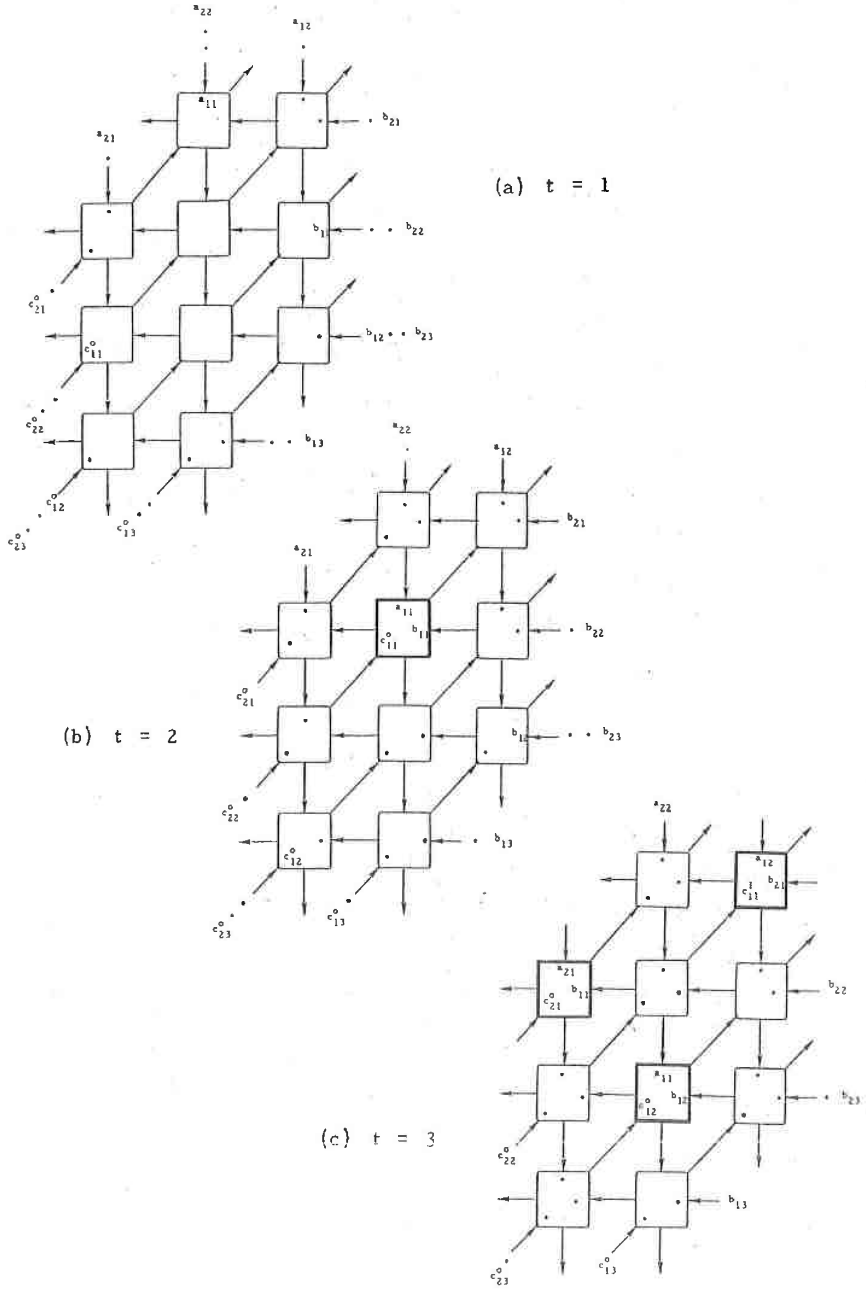


Figure 2.11. Réseau hexagonal



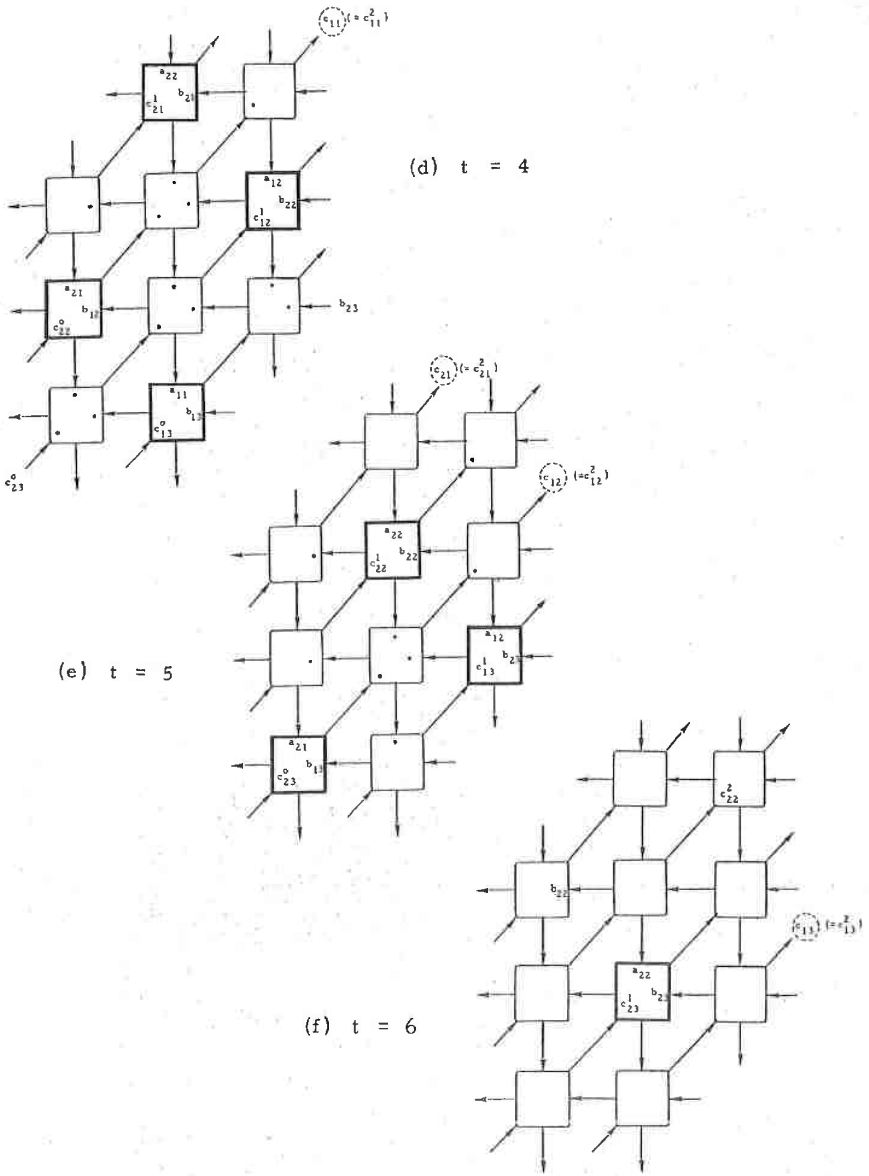


Figure 2.12. Simulation d'exécution du réseau hexagonal

IV. Conclusion

De nombreux problèmes peuvent être traités par des réseaux systoliques. Pour un problème donné, il existe un ensemble de réseaux systoliques adaptés à sa résolution. Tous n'ont pas les mêmes caractéristiques concernant la nature des cellules, le temps total d'exécution, le nombre de cellules, le "taux" de parallélisme (rapport entre le nombre de cellules actives à un instant donné et le nombre total de cellules), etc.

Notons que le nombre de cellules d'un réseau, associé à un problème donné, dépend des bornes du problème. Ainsi, pour le produit de convolution, le nombre de cellules d'un réseau est une fonction des données n et m. Cette fonction diffère d'un réseau à l'autre. Ceci est une limitation non négligeable : un réseau systolique donné est adapté à la résolution d'un problème unique dont les bornes sont "figées". Néammoins, si l'on sait construire un réseau pour un problème défini par certaines de ces bornes, on sait également en construire un pour des valeurs différentes des bornes. La forme des deux réseaux est la même, seul le nombre de cellules est différent.

Nous proposons aux chapitres 3 et 4 une méthode permettant de construire un ensemble de réseaux systoliques associés à un problème donné. Le logiciel, basé sur cette méthode, dont nous exposons la structure au chapitre 5, permet de déterminer les réseaux associés à un problème dont les bornes sont fixées.

CHAPITRE 3

PRÉSENTATION INTUITIVE DE LA MÉTHODE

Notre objectif est de proposer une méthode constructive permettant de déterminer, à partir de l'énoncé d'un problème sous forme d'équations récurrentes, un ensemble de réseaux systoliques solution de ce problème. Les réseaux systoliques obtenus n'ont pas tous les mêmes caractéristiques : ils peuvent être systoliques purs ou semi-systoliques avec diffusion; leurs cellules peuvent être fonctionnelles ou à mémoire; le nombre de cellules et le temps total d'exécution varient d'un réseau à l'autre; etc. L'intérêt d'obtenir un ensemble de réseaux est de permettre à l'utilisateur de choisir le réseau qui lui convient en fonction de ses critères personnels, qui peuvent porter sur la nature des cellules, la minimisation du nombre de cellules ou du temps d'exécution, etc.

Nous ne nous intéressons ici qu'aux *problèmes* dits *systolisables*, c'est-à-dire aux problèmes pour lesquels il existe au moins un algorithme systolique correspondant.

Dans ce chapitre, nous décrivons de manière intuitive les idées générales de la méthode, tandis que dans le chapitre 4, nous en faisons une présentation complète et formelle.

Décrivons les principaux points de la méthode que l'on retrouve dans les deux chapitres.

— La première étape consiste à déterminer une spécification du problème à partir des équations.

— Une représentation du problème est un ensemble de points de coordonnées entières de $\mathbf{R}^r$. Un point correspond à un pas d'un calcul de récurrence, auquel est associée une abscisse temporelle qui définit l'instant où ce calcul est effectué. A un problème systolisable donné sont associés un ensemble de représentations différentes. Elles sont obtenues, à partir d'une représentation initiale, par application d'une suite de transformations affines. Le but de la deuxième étape est de déterminer l'ensemble des représentations associées à un problème.

— Une représentation du problème définit l'ensemble des calculs élémentaires à réaliser et l'instant auquel chaque calcul doit être effectué. La

dernière étape consiste donc à définir, pour chaque calcul élémentaire, sur quelle cellule du réseau systolique correspondant il doit être effectué. Plusieurs répartitions différentes des calculs élémentaires sur les cellules d'un réseau sont possibles. On détermine donc, pour une représentation, un ensemble de "fonctions d'allocation" des cellules. Chacune de ces fonctions définit sur quelle cellule s'effectue chacun des calculs élémentaires. On peut alors déduire de cette fonction et de la représentation, le réseau systolique correspondant.

Nous illustrons cette présentation avec l'exemple du produit de convolution décrit au chapitre 2. Le problème est défini par le système d'équations de récurrence :

$$(1)\begin{cases} y_i^k = y_i^{k-1} + w_l \times x_{i+l}, & k = 0,\ldots,m, \quad i = 0,\ldots,n-m \\ y_i^{-1} = 0 \end{cases}$$

Nous nous limitons ici au choix $l = k$ correspondant au calcul de récurrence dans l'ordre croissant de l'indice k :

$$y_i = w_0 \times x_i + w_1 \times x_{i+1} + \cdots + w_m \times x_{i+m}$$

Dans la suite, nous prenons $n = 7$ et $m = 2$.

I. Le domaine et la spécification du problème

Les équations récurrentes associées au problème définissent un ensemble d'opérations élémentaires indicées par un r-uplet. Nous appelons l'ensemble de ces r-uplets le *domaine D* associé au problème. Nous le représentons dans un repère orthonormé de $\mathbf{R}^r$ par des points de coordonnées entières.

Pour le produit de convolution, une opération élémentaire de la forme $y_i^k = y_i^{k-1} + w_k \times x_{i+k}$ est indicée par le couple (i,k), et le domaine associé au problème est :

$$D = \{(i,k) \in \mathbb{Z}^2 \quad | \quad 0 \leq i \leq n-m, \quad 0 \leq k \leq m\}$$

Il est représenté dans $\mathbf{R}^2$ muni de la base orthonormée $(\vec{i},\vec{k})$ comme le montre la figure 3.1 (avec $n = 7$ et $m = 2$).

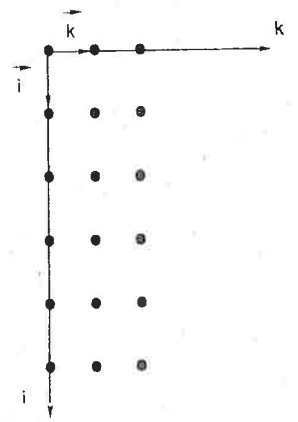


FIGURE 3.1. Représentation du domaine D

En chaque point de D, l'opération correspondant à ce point est caractérisée par la fonction réalisée et par la liste des données utilisées par cette opération.

Pour le produit de convolution, la fonction réalisée est la même en tout point (i,k) du domaine D :

$$y_i^k = f(y_i^{k-1}, w_k, x_{i+k}) = y_i^{k-1} + w_k \times x_{i+k}$$

La liste des données utilisées au point (i,k) est la liste des paramètres de cette fonction , c'est-à-dire ici y_i^{k-1}, w_k et x_{i+k}. Ceci est illustré par la figure 3.2 où la fonction réalisée est omise car elle est la même en tout point du domaine. Elle est de la forme $f(a,b,c) = a + b \times c$ avec a,b,c réels.

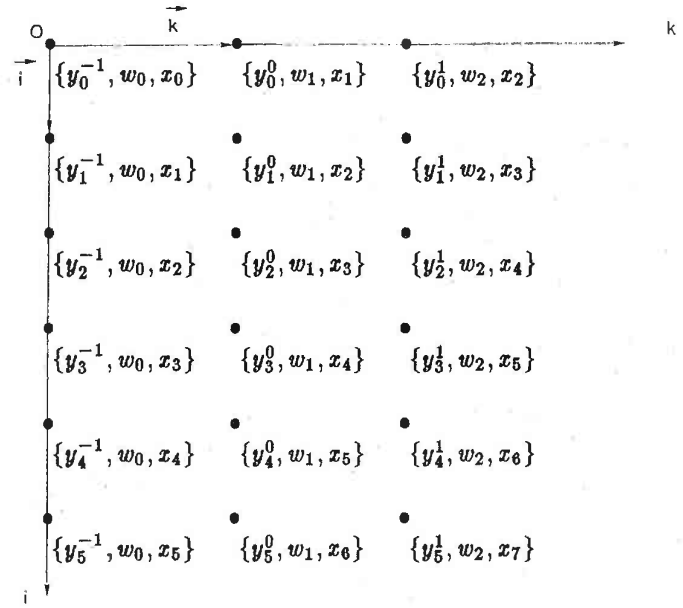


FIGURE 3.2. Représentation du problème

Le domaine D, la liste des données et la fonction réalisée en chaque point de D forment la *spécification systolique* du problème.

On peut retrouver les équations récurrentes à partir de cette spécification. La fonction réalisée aux points du domaine définit la forme de la récurrence, le domaine définit les bornes des indices de cette récurrence.

II. L'espace-temps et la base canonique

Une caractéristique des systèmes d'équations récurrentes que l'on veut résoudre par un réseau semi-systolique avec diffusion ou par un réseau systolique pur est :

— un résultat élémentaire $y_i = y_i^m$ est calculé par accumulations successives de ces éléments de récurrence $y_i^k, k = 0, \ldots, m$, ces différents y_i^k étant calculés à des instants successifs.
— des résultats élémentaires y_i et y_j différents peuvent être calculés simultanément, c'est-à-dire que deux calculs quelconques de récurrence y_i^k et $y_j^{k'}$ peuvent être simultanés.

Nous introduisons la notion de temps pour établir une relation d'ordre temporel entre les calculs de récurrence. Pour prendre en compte le fait que cet ordre est total sur les calculs de récurrence d'un même résultat élémentaire (comme l'exprime la première remarque ci-dessus), l'axe de récurrence $(O, \vec{k})$ est choisi pour axe des temps. Nous définissons ainsi un nouveau repère orthonormé, appelé *espace-temps*.

L'espace-temps pour le produit de convolution est le repère $(O', \vec{i}, \vec{t})$ avec $O' = O$ et $\vec{t} = \vec{k}$, comme l'illustre la figure 3.3.

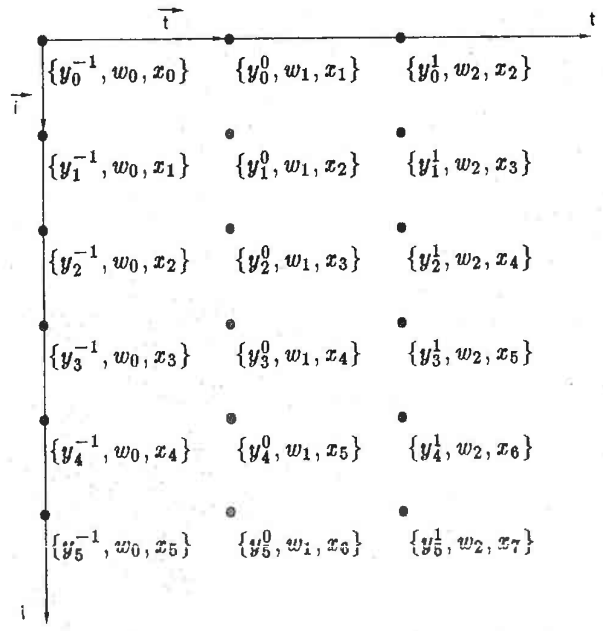


FIGURE 3.3. Représentation du problème dans l'espace-temps $(\vec{t} = \vec{k})$

Lorsque l'on choisit pour le système d'équations (1) $l = m - k$, c'est que l'on souhaite exécuter une récurrence élémentaire dans l'ordre décroissant de l'indice k :

$$y_i = w_m \times x_{i+m} + w_{m-1} \times x_{i+m-1} + \cdots + w_0 \times x_i$$

Dans ce cas, l'espace-temps doit être défini par $\vec{t} = -\vec{k}$ comme le montre la figure 3.4.

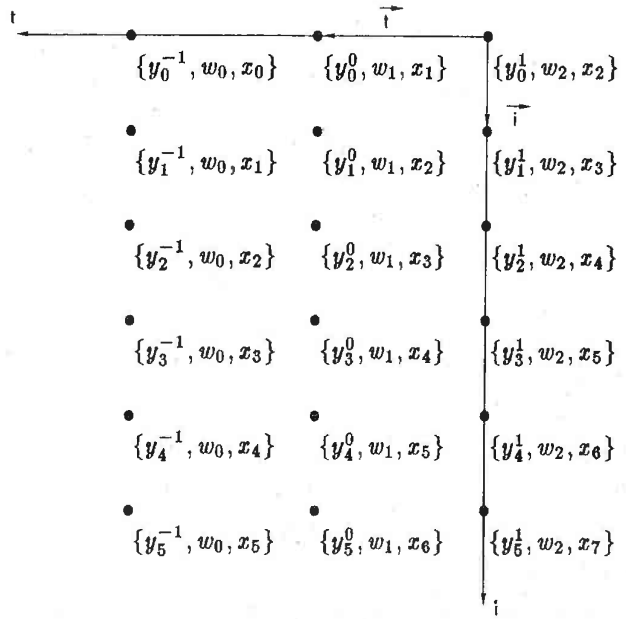


FIGURE 3.4. Deuxième choix possible pour l'axe des temps $(\vec{t} = -\vec{k})$

Une représentation du problème définit un certain ordonnancement des calculs élémentaires. Chaque point d'une représentation, correspondant à un calcul élémentaire, est caractérisé par le r-uplet de ses indices de calcul et par une abscisse temporelle, coordonnée du point sur l'axe $(O', \vec{t})$ de l'espace-temps, qui représente l'instant où le calcul s'exécute. Les indices du calcul associé à un point sont les coordonnées de ce point dans une base

$(\vec{b}_i, \vec{b}_k)$ appelée *base canonique*. Sur la représentation de la figure 3.1, cette base $(\vec{b}_i, \vec{b}_k)$ coïncide avec la base orthonormée $(\vec{i}, \vec{k})$ de $\mathbf{Z}^2$.

REMARQUE. — Notons que les réseaux semi-systoliques avec collecteur sont caractérisés par le fait que tous les calculs de récurrence $y_i^k = y_i^{k-1} + w_l \times x_{i+l}, k = 0, \ldots, m$, d'un résultat élémentaire y_i sont exécutés simultanément. Ainsi pour le produit de convolution, à un instant t donné, les m produits $w_0 \times x_i, w_1 \times x_{i+1}, \ldots, w_m \times x_{i+m}$ correspondants au calcul du résultat élémentaire y_i sont effectués simultanément sur m celllules différentes. Les m résultats sont alors envoyés au collecteur qui les additionne.

Pour prendre en compte cette simultanéité d'exécution, l'axe des temps doit être choisi orthogonal à l'axe des récurrences $(O, \vec{k})$. Ainsi pour le produit de convolution, il faudrait prendre l'axe $(O, \vec{i})$ pour axe des temps, comme le montre la figure 3.5. Les seules ressources utilisées en un point (i, k) de la représentation sont w_k et x_{i+k}; d'où la disparition du y_i^{k-1} de la liste des ressources associées.

Un réseau avec collecteur correspondant est représenté figure 3.6. A l'instant $t = 1$, les 3 cellules sont actives. Elles calculent chacune un produit du calcul de récurrence de y_0, puis envoient ces 3 résultats y_0^0, y_0^1, y_0^2 au collecteur qui les additionne pour restituer la valeur de y_0. A l'instant suivant, le même processus est répété pour le résultat y_1, et ainsi de suite jusqu'à y_{n-m}.

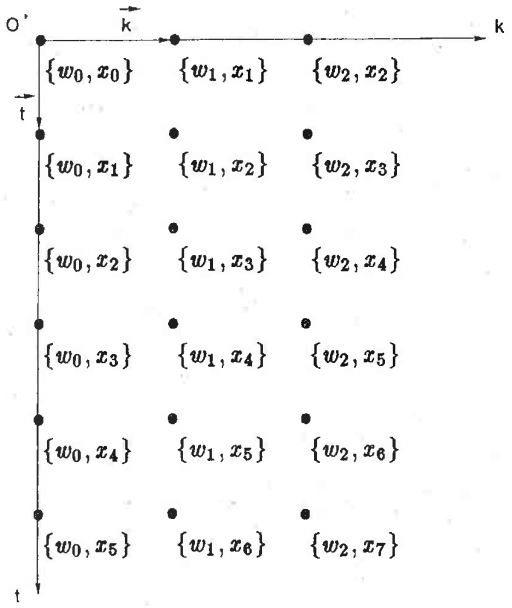


FIGURE 3.5. Choix de l'espace-temps pour un réseau semi-systolique avec collecteur

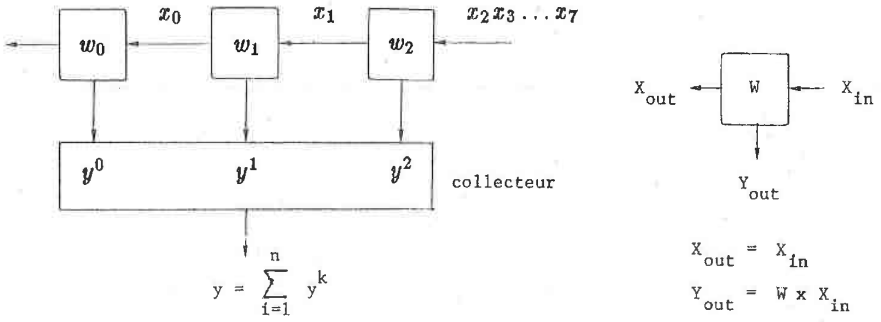


FIGURE 3.6. Réseau semi-systolique avec collecteur

III. Transformations appliquées à une représentation

Une deuxième caractéristique des systèmes d'équations récurrentes considérés ici, est qu'une donnée intervient dans plusieurs calculs élémentaires. Ainsi pour le produit de convolution, un w_k donné est utilisé dans $n - m$ calculs élémentaires de la forme $w_k \times x_{i+k}$ associés chacun à un résultat élémentaire y_i différent.

Cette caractéristique peut être réalisée sur un réseau systolique de deux façons :

— les calculs élémentaires utilisant les occurrences d'une même donnée sont effectués simultanément sur des cellules différentes. La donnée considérée doit donc être disponible en même temps sur toutes ces cellules. Pour cela, elle est diffusée. On obtient ainsi des réseaux semi-systoliques avec diffusion.

— les calculs élémentaires utilisant les occurrences d'une donnée sont effectués à des instants successifs. Ces calculs peuvent être réalisés soit sur la même cellule, soit sur des cellules adjacentes. Dans le premier cas, la donnée est donc utilisée à des instants successifs sur la même cellule. Ceci correspond à des réseaux systoliques composés de cellules fonctionnelles munies d'une zone de mémoire locale constante contenant la donnée considérée. Dans le deuxième cas, la donnée est utilisée à des instants successifs sur des cellules contiguës. Cela signifie que la donnée doit circuler dans le réseau de cellules en cellules. Les réseaux ainsi obtenus sont donc systoliques purs.

Notre **objectif** est d'obtenir un **ensemble de réseaux systoliques** solution d'un problème donné. Ces réseaux ont des caractéristiques différentes. En particulier, l'ordre et la simultanéité des calculs élémentaires peuvent varier d'un réseau à l'autre.

Une représentation du problème dans l'espace-temps définit un certain ordre de ces calculs. Pour obtenir des réseaux où l'ordre est différent, il faut donc pouvoir considérer une autre représentation du problème. Pour cela, nous appliquons des transformations permettant, à partir d'une représentation donnée, d'en obtenir de nouvelles. Nous définissons ainsi un ensemble d'ordonnancements des calculs élémentaires, et pour chacun de ces ordonnancements, nous pouvons déterminer un ensemble de réseaux systoliques.

Or l'ordre des calculs peut être caractérisé par la façon dont sont utilisées les données intervenant dans différents calculs. Modifier l'ordre des calculs consiste donc à modifier la façon dont sont utilisées ces données. C'est pourquoi les transformations que nous allons appliquer dépendent de conditions sur l'utilisation des données à usage multiple.

Appliquer une transformation sur une représentation consiste à modifier l'abscisse temporelle des points, afin de modifier l'ordre et la simultanéité des calculs, sans pour autant modifier le r-uplet associé aux points. En effet, ce r-uplet représente les coordonnées du calcul asssocié au point, qui sont invariantes quelle que soit la représentation. C'est pourquoi les seules transformations possibles sont celles dont l'effet est de déplacer les points du domaine D parallèlement à l'axe des temps $(O', \vec{t})$.

Du point de vue réalisation d'un réseau systolique par un circuit VLSI, les réseaux les plus interessants sont les réseaux systoliques purs dans lesquels il n'y a pas de communication globale de données (comme par exemple des diffusions), c'est-à-dire où toutes les données utilisées plusieurs fois le sont à des instants successifs. C'est pourquoi les seules transformations étudiées ici ont pour but de transformer une représentation munie d'une utilisation simultanée d'une donnée en une représentation où cette donnée est alors utilisée successivement.

La condition d'application d'une telle transformation sur une représentation R d'un problème peut donc s'exprimer ainsi : "il existe pour la représentation R, une donnée d utilisée simultanément en différents points du domaine D".

Soyons plus précis et notons (D_t), l'ensemble des points du domaine de même abscisse temporelle t pour une représentation donnée. Ce sont des segments de droite parallèles à l'axe $(O', \vec{i})$ dans l'espace-temps. Une transformation sera appliquée sur cette représentation si la condition d'utilisation simultanée des occurrences d'une donnée est observée, c'est-à-dire s'il existe au moins une abscisse temporelle t_0 telle que plusieurs points de D_{t_0} contiennent cette donnée dans leur liste associée.

Appliquer une transformation consiste à supprimer la simultanéité d'utilisation des occurrences de la donnée en imposant leur utilisation successive dans les différents calculs de manière régulière. Cela signifie que l'image

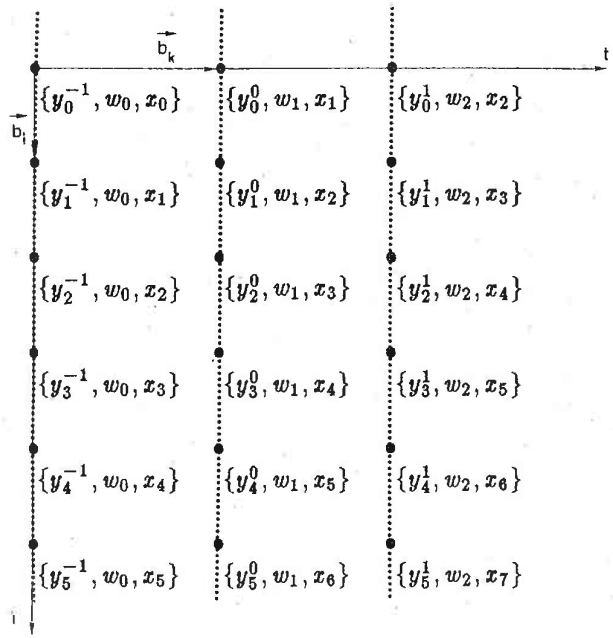


FIGURE 3.7. Représentation initiale dans l'espace-temps avec $\vec{t} = \vec{k}$

de toute droite D_t par cette transformation est aussi une droite dans la représentation image.

Ainsi pour la représentation initiale du produit de convolution dans l'espace-temps (cf. figure 3.7), les droites (D_t) sont représentées en pointillé. On constate que les occurrences d'une donnée $w_k, k = 0, \ldots, m$, sont utilisées simultanément en tous points de la droite D_t où $t = k$. On peut donc appliquer une transformation qui associe à toute droite D_t parallèle à l'axe $(O', \vec{i})$ dans l'espace-temps, une droite non parallèle à l'axe $(O', \vec{i})$.

On distingue deux types de transformations conduisant à des droites images différentes :

— une transformation, appelée T_c, pour laquelle les droites images sont de pente $+p$ (cf. figure 3.8 (a)).

— une transformation, appelée T_d, pour laquelle les droites images sont de pente $-p$ (cf. figure 3.8 (b)).

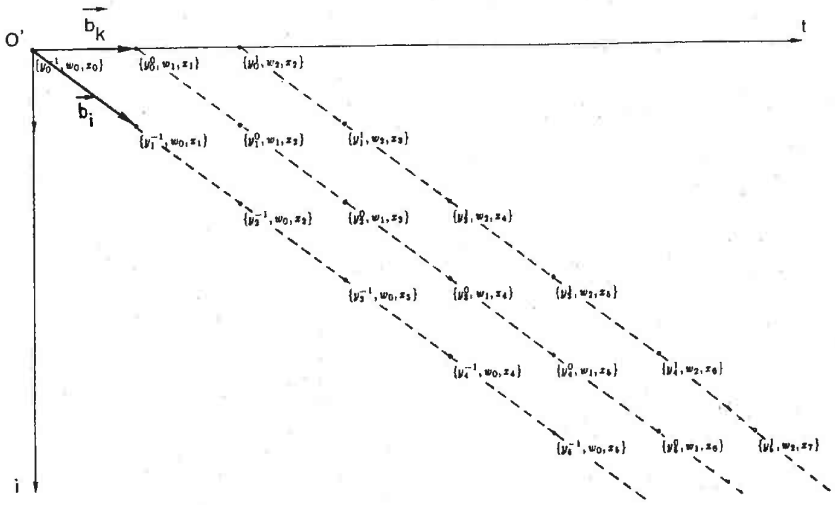
où p est le plus petit entier positif non nul tel que l'abscisse temporelle des points du domaine, après transformation, soit entière. La valeur de p est selon les cas 1 ou $1/\tau$, τ étant le nombre entier égal à l'ordonnée du vecteur $\vec{b}_k$ dans la base de l'espace-temps $(\vec{i}, \vec{t})$. τ est supérieur ou égal à 1 et représente intuitivement l'espacement des droites D_k formées de l'ensemble des points du domaine de même indice de récurrence k.

REMARQUE. — On désigne les deux types de transformations par T_c et T_d pour indiquer que la pente des droites images est croissante $(+p)$ pour la transformation T_c ou décroissante $(-p)$ pour une transformation T_d.

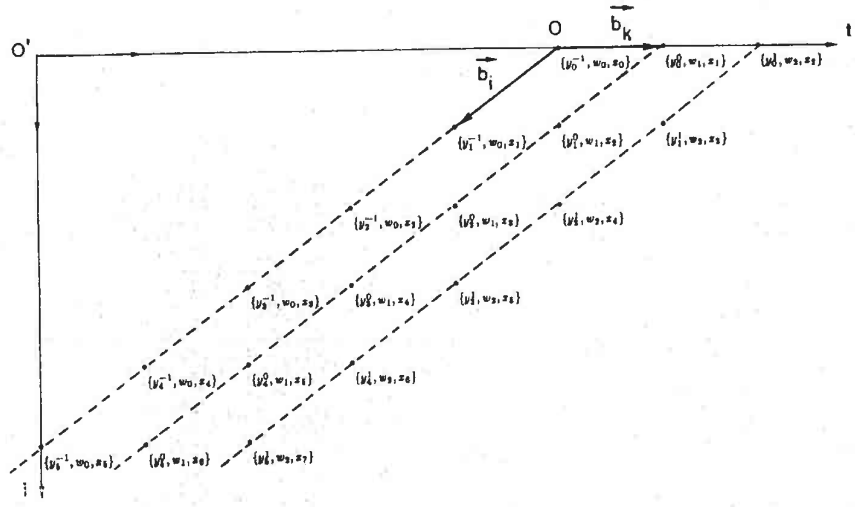
On se limite aux transformations pour lesquelles les droites images sont de pente $+p$ ou $-p$, avec $p = 1$ ou $p = 1/\tau$, car ce sont les "meilleures". En effet, appliquer une transformation supprime la simultanéité d'utilisation des occurrences d'une donnée mais augmente le temps total d'exécution des calculs. Ainsi, pour la représentation initiale (cf. figure 3.7), le temps total d'exécution est $t = m + 1 = 3$ unités de temps, tandis que pour la représentation issue d'une transformation T_c (cf. figure 3.8 (a)), ce temps est de $t = m + (n - m) + 1 = 8$ unités de temps. Lorsque la pente des droites images est $\pm p$, deux points d'une même droite initiale D_t d'abscisses consécutives sur $\vec{i}$, ont pour image deux points dont les ordonnées sur $\vec{t}$ différent de 1. Cela signifie que ces deux calculs simultanés sont transformés en deux calculs immédiatement successifs. Considérer des transformations telles que la pente des droites obtenues soit $+q$ ou $-q$, avec $q \rangle p$ et $q \in \mathbf{Q}$, aurait pour conséquence d'augmenter encore davantage le temps d'exécution ce qui est sans intérêt dans le cadre d'une exécution parallèle, car le nombre de calculs simultanés serait réduit.

La figure 3.8 montre les représentations images de la représentation initiale par une transformation T_c (figure (a)) ou T_d (figure (b)). Les droites images des droites (D_t) de la représentation initiale sont représentées en pointillé. Elles sont de pente $+1$ dans le premier cas, et de pente -1 dans le second.

Considérons maintenant la représentation de la figure 3.8 (b). Sur cette représentation, aucune simultanéité d'utilisation d'occurrences d'une même donnée n'est observée. En effet, les points d'une droite D_t parallèle à l'axe $(O', \vec{i})$ utilisent tous des données différentes. Aucune nouvelle transformation n'est donc appliquée sur cette représentation.



(a) Transformation T_c



(b) Transformation T_d

FIGURE 3.8. Application d'une transformation sur la représentation initiale

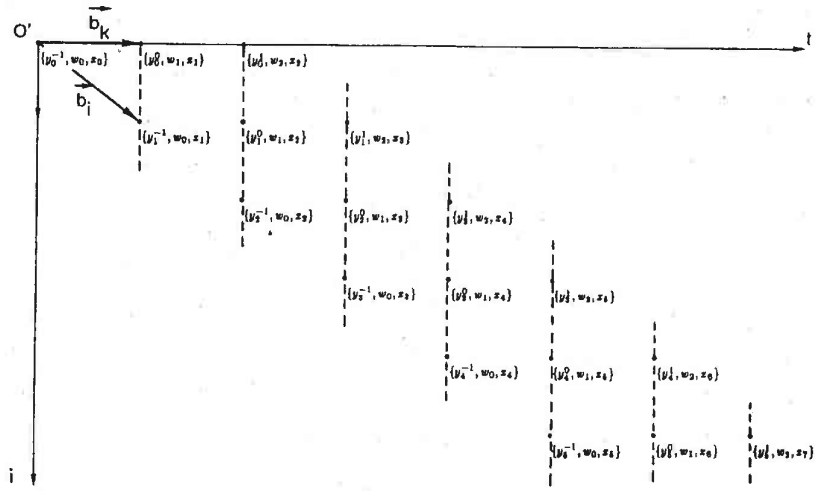
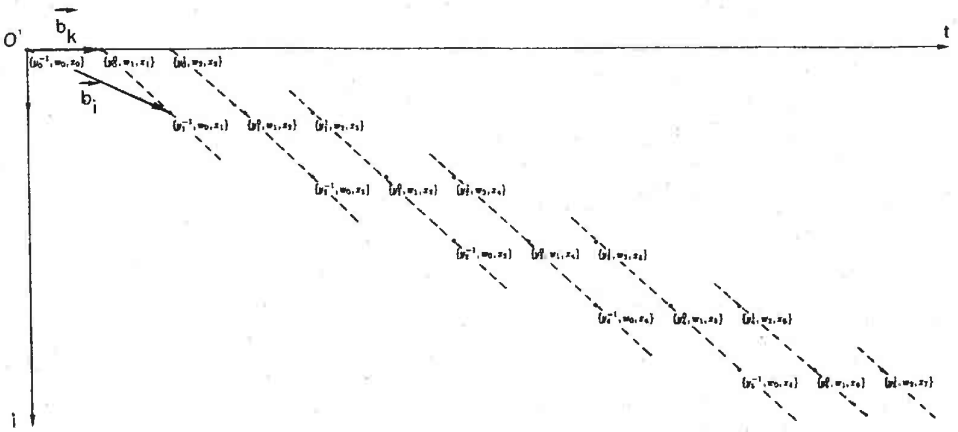


FIGURE 3.9. Une représentation associée au produit de convolution

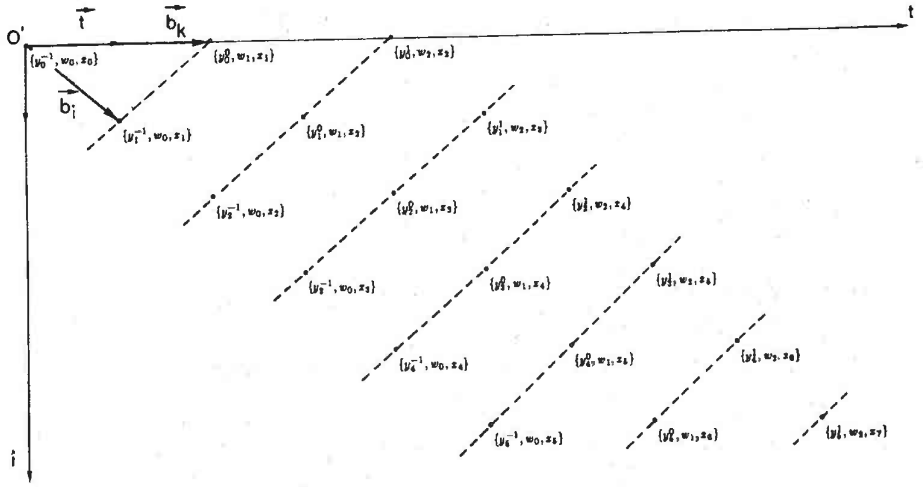
Par contre, la représentation de la figure 3.8 (a) définit une condition d'utilisation simultanée d'une donnée. En effet, comme le montre la figure 3.9, les points de chacune des droites D_t (en pointillé) utilisent la même donnée x_t.

Nous pouvons donc appliquer sur cette représentation une transformation soit de type T_c, soit de type T_d. Les représentations ainsi obtenues sont visualisées figure 3.10. L'image des droites (D_t), en pointillé, sont de pente $+1$ pour la transformation T_c et -1 pour la transformation T_d $(\tau = 1)$.

Sur les deux nouvelles représentations obtenues, aucune condition d'utilisation simultanée de données n'est observée. On n'applique donc pas de nouvelles transformations.



(a) Transformation T_c



(b) Transformation T_d

FIGURE 3.10. Représentations images de celle de la figure 3.9

Les transformations étudiées ici sont des transformations affines dont l'effet est de donner pour image aux droites (D_t), parallèles à l'axe $(O', \vec{i})$ dans l'espace-temps, des droites de pente $\pm p$, avec $p = 1$ ou $p = 1/\tau$. Notons que leur définition dépend de la nature de la première transformation appliquée à la représentation initiale, comme cela peut être constaté sur l'exemple traité ici.

En effet, prenons l'exemple d'une transformation T_d. S'il s'agit de la première transformation appliquée à la représentation initiale (comme sur la figure 3.8 (b)), ou si la première transformation est de même type, la transformation T_d est définie par :

$$\begin{array}{rccc} T_d : & \mathbf{N}^2 & \rightarrow & \mathbf{N}^2 \\ & (i,t) & \rightarrow & (i, t - i + i_{max}) \end{array}$$

où i_{max} est la borne supérieure de l'indice i sur le domaine, ici $i_{max} = 5$. Dans ce cas, les droites images sont de pente ± 1.

Si la première transformation appliquée à la représentation initiale est de type contraire, T_c (comme sur la figure 3.10 (b)), alors la transformation T_d est définie par :

$$\begin{array}{rccc} T_d : & \mathbf{N}^2 & \rightarrow & \mathbf{N}^2 \\ & (i,t) & \rightarrow & (i, t + k) \end{array}$$

où k est l'ordonnée du point (i,t) dans la base canonique $(\vec{b_i}, \vec{b_k})$ du problème, c'est-à-dire l'indice de récurrence du calcul élémentaire correspondant. Dans ce cas, les droites images sont de pente $\pm 1/\tau$.

Comme la représentation de la figure 3.9 est telle que $\tau = 1$, les droites images des droites D_t par la transformation T_d sont de pente $-1/\tau = -1$ (figure 3.10 (b)).

Une représentation d'un problème est caractérisée par la base canonique exprimée dans l'espace-temps. La base canonique définit les r-uplets du domaine D. Un point (i,k) dans cette base représente le calcul élémentaire correspondant, $y_i^k = y_i^{k-1} + w_k \times x_{i+k}$.

Une représentation peut donc être définie par la matrice de passage P de la base canonique à l'espace-temps et par le vecteur de translation $V = \overrightarrow{OO'}$ exprimé dans l'espace-temps. O (resp. O') est l'origine du repère associé à la base canonique (resp. à l'espace-temps).

Examinons l'exemple du produit de convolution présenté ci-dessus, où la base canonique $(\vec{b_i}, \vec{b_k})$ est exprimée dans l'espace-temps muni du repère $(O', \vec{i}, \vec{t})$.

Pour la représentation initiale du problème (cf. figure 3.7), la base canonique est égale à la base $(\vec{i}, \vec{t})$ de l'espace-temps, la matrice de passage P est donc la matrice identité et le vecteur V est nul. Pour les différentes représentations on a :

$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ et $V = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ pour la figure 3.8(a)

$P = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$ et $V = \begin{pmatrix} 0 \\ 5 \end{pmatrix}$ pour la figure 3.8(b)

$P = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$ et $V = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ pour la figure 3.10(a)

$P = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}$ et $V = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ pour la figure 3.10(b)

IV. Réseaux systoliques associés à une représentation

La dernière étape de la démarche consiste à déterminer, pour une représentation donnée du problème, des réseaux systoliques correspondants. Une représentation définit un certain ordonnancement des calculs simultanés, indépendamment des cellules. Il faut maintenant choisir comment ces calculs sont répartis sur les différentes cellules du réseau.

Pour cela, nous faisons le choix, sur la représentation considérée, d'une "direction d'allocation" $\vec{\xi} \in \mathbf{R}^r$. Pour le produit de convolution, on a $r = 2$. Cette direction définit un ensemble de droites de l'espace $\mathbf{R}^2$. Tous les points du domaine D appartenant à une de ces droites représentent les calculs effectués sur la même cellule.

Notons que tout vecteur de $\mathbf{R}^r$ ne peut être une direction d'allocation. En particulier, $\vec{\xi}$ ne peut être choisi perpendiculaire au vecteur $\vec{t}$ de l'espace-temps car dans ce cas, tous les points d'une même droite définie par $\vec{\xi}$ ont même abscisse temporelle. Par définition de $\vec{\xi}$, ils devraient être exécutés sur la même cellule en même temps, ce qui est impossible, une cellule ne pouvant exécuter qu'un calcul au plus à un instant donné.

Les points d'une droite définie par la direction d'allocation $\vec{\xi}$ ont donc des abscisses temporelles différentes. Les calculs correspondants à ces points

seront exécutés sur une même cellule à des instants différents, dans l'ordre imposé par les abscisses temporelles.

A partir de cette direction $\vec{\xi}$, nous déterminons la "fonction d'allocation" a de $D \subset \mathbf{R}^r$ dans $\mathbf{R}^{r-1}$: elle indique la référence de la cellule (repérée par $r-1$ indices) où le calcul d'un point de D s'effectue. La démarche de déduction du réseau systolique à partir de la fonction d'allocation et de la représentation du problème est explicitée dans le chapitre 4. Nous en donnons quelques idées sur les deux exemples ci-dessous.

EXEMPLE 1. — Considérons la représentation du problème obtenue à la figure 3.8 (a). Si nous choisissons comme direction d'allocation le vecteur $\vec{\xi} = (1,0)_{B.C.}$ exprimé dans la base canonique ou $\vec{\xi} = (1,1)_{E.T.}$ exprimé dans l'espace-temps, nous obtenons un réseau systolique avec 3 cellules comme l'indique la figure 3.11 où les points des droites correspondant aux calculs effectués sur la même cellule sont entourés d'un trait pointillé.

On constate sur cette figure que :
— tous les points correspondant à des calculs effectués sur une même cellule de numéro k, contiennent, dans la liste des données associées, le même w_k. Cette donnée est donc utilisée pour tous les calculs effectués sur cette cellule : elle est constante de la cellule considérée. Le réseau est alors constitué de cellules fonctionnelles munies d'une zone de mémoire locale constante contenant une donnée w_k.
— tous les points ayant même abscisse temporelle t contiennent dans la liste des données associées le même x_t. Chaque donnée de la suite $(x_i)_{i=0,\ldots,n}$ est donc utilisée simultanément dans toutes les cellules concernées : il y a diffusion des données de cette suite dans l'ordre croissant des indices $x_1, x_2, x_3 \ldots$, car la donnée x_i est utilisée à l'instant $t = i$, comme le montre la figure 3.11.
— une donnée élémentaire y_i^{-1} est utilisée par un point associé à la cellule numéro 0, dont le calcul s'exécute à l'instant t, tandis que y_i^0 est utilisée par un point, associé à la cellule numéro 1 dont le calcul s'exécute à l'instant $t+1$ et que y_i^1 est utilisée par un point, associé à la cellule numéro 2, dont le calcul s'exécute à l'instant $t+2$. Chaque donnée de la suite $(y_i)_{i=,0,\ldots,n-m}$ circule donc dans le réseau de la cellule numéro 0 vers la cellule numéro 2.

Ces constatations se traduisent par le réseau de la figure 3.12.

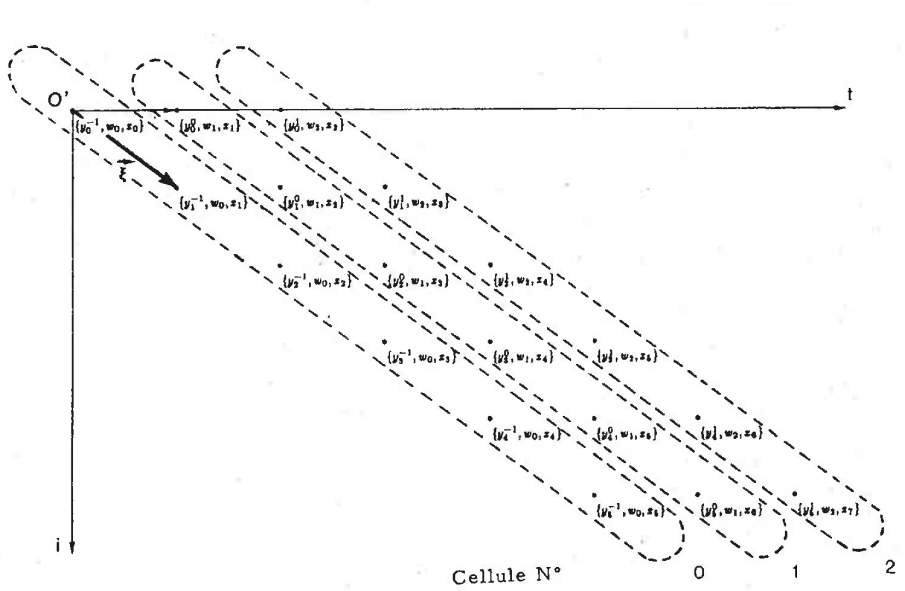


FIGURE 3.11. Choix de la direction d'allocation $\vec{\xi} = (1,0)_{B.C.} = (1,1)_{E.T.}$

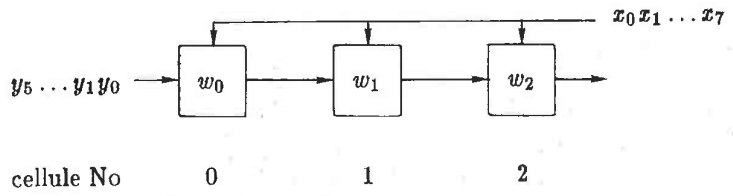


FIGURE 3.12. Réseau correspondant à la représentation de la figure 3.11

EXEMPLE 2. — Sur la représentation de la figure 3.8 (a), nous choisissons maintenant comme direction d'allocation $\vec{\xi} = (0,1)_{B.C.} = (0,1)_{E.T.}$. Cette direction définit un ensemble de droites. Les points d'une même droite, entourés de pointillés sur la figure 3.13, représentent les calculs exécutés sur la même cellule.

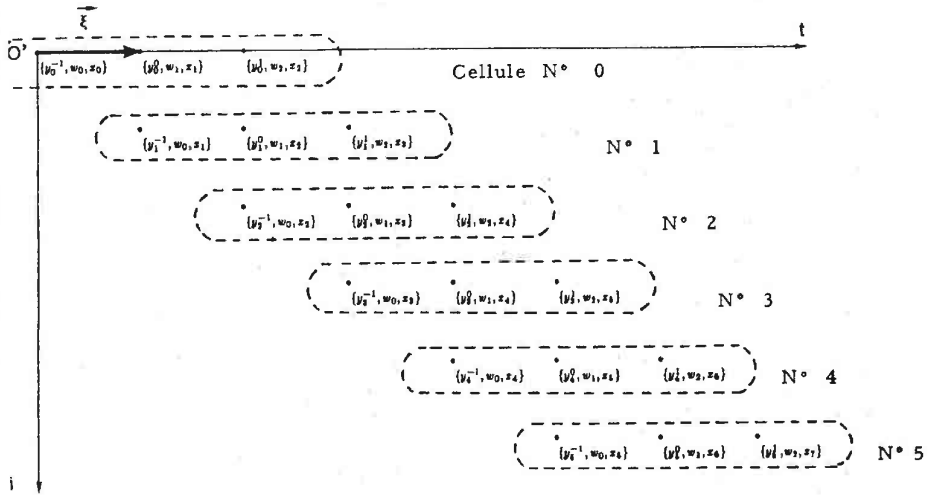


FIGURE 3.13. Choix de la direction d'allocation$\vec{\xi} = (0,1)_{B.C.}$

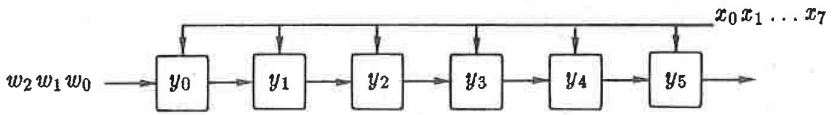


Figure 3.14. Réseau correspondant à la représentation de la figure 3.13

On constate sur cette figure que :
— pour i fixé, tous les $y_i^k, k = -1, \ldots, 1$, sont utilisés sur une même cellule. Cela conduit donc à un réseau composé de cellules à mémoire. Un résultat élémentaire y_i est calculé par la i-ème cellule dans sa zone mémoire utilisée comme variable.
— comme sur l'exemple 1, les données $(x_i)_{i=0,\ldots,n}$ sont diffusées simultanément à toutes les cellules du réseau.
— une donnée w_k, k fixé, est utilisée par la cellule numéro 0 à l'instant $t = k$, puis par la cellule numéro 1 à l'instant $t + 1$ et ainsi de suite jusqu'à la cellule numéro 5. Chaque donnée de la suite $(w_k)_{k=0,\ldots,m}$

circule donc à travers les cellules, de celle numéro 0 vers celle numéro 5.
Le réseau correspondant est représenté figure 3.14.

Notons que le choix de la direction d'allocation $\vec{\xi}$ détermine la nature des cellules du réseau correspondant. En effet, si l'on choisit $\vec{\xi}$ parallèle au vecteur $\vec{t}$ de l'espace-temps, c'est-à-dire si $\vec{\xi} = (0,n)_{E.T.}, n \in \mathbf{Z}$, le réseau systolique correspondant est constitué de cellules à mémoire (cf. figure 3.14). Dans les autres cas, le réseau est constitué de cellules fonctionnelles. Le choix $\vec{\xi} = (0,n)_{E.T.}$ n'est évidemment possible que si les bornes du domaine sont finies. Dans le cas contraire, un tel choix nécessiterait une infinité de cellules.

Les réseaux présentés au chapitre 2 sont obtenus en appliquant la démarche esquissée ci-dessus.

CHAPITRE 4

PRÉSENTATION FORMELLE DE LA MÉTHODE

Nous présentons maintenant notre méthode de conception d'algorithmes systoliques de manière formelle.

Les problèmes pris en compte par cette méthode sont les problèmes dits ***systolisables***, pour lesquels il existe au moins un algorithme systolique. Les énoncés qui caractérisent ces problèmes sont définis par des systèmes d'équations récurrentes de certaines formes.

Définir un problème systolisable, c'est donc déterminer un système d'équations récurrentes énoncé de ce problème. Nous décrivons au paragraphe I quelle forme peut avoir un tel système pour être l'énoncé d'un problème systolisable, puis nous proposons une méthode pour déterminer de tels systèmes. Cette méthode est basée sur la décomposition du problème en modules.

Nous définissons au paragraphe II, la spécification systolique d'un problème et sa représentation dans $\mathbf{R}^r$.

Au paragraphe III, nous définissons les transformations affines applicables sur une représentation et les conditions méthodologiques sur les données du problème permettant leur application.

Au paragraphe IV, nous définissons une relation sur les représentations. Elle permet de déterminer les représentations ayant des réseaux associés "similaires".

Au paragraphe V, nous montrons comment déterminer pour chaque représentation une famille de fonctions d'allocation et le réseau systolique associé à chaque fonction.

Notons que dans la pratique, l'énoncé d'un problème systolisable s'exprime soit dans $\mathbf{R}^2$ (réseaux systoliques linéaires), soit dans $\mathbf{R}^3$ (réseaux bidimensionnels).

I. Enoncé d'un problème systolisable

Un problème est systolisable s'il peut s'exprimer par un énoncé d'une certaine forme. Nous décrivons quelle peut être la forme d'un tel énoncé et nous proposons une méthode de décomposition du problème permettant de l'obtenir.

I.1 Forme de l'énoncé. — Un problème systolisable est défini par un énoncé dans $\mathbf{R}^2$ quand il est résolu par un réseau linéaire et dans $\mathbf{R}^3$ quand il est résolu par un réseau bidimensionnel. La forme de l'énoncé est donnée pour les problèmes dans $\mathbf{R}^3$, les simplifications pour $\mathbf{R}^2$ sont évidentes.

Un énoncé d'un problème systolisable défini dans $\mathbf{R}^3$ est décrit par un système d'équations récurrentes dépendant de deux indices. Il est de la forme :

$$(1)\begin{cases} y_{ij}^k = f(y_{ij}^{k-1}, a^1, \ldots, a^u) & a < k \leq b, i \in I, j \in J \\ y_{ij}^a = v & v \in \mathbf{R} \end{cases}$$

où $a^1, \ldots, a^u$ sont des données du problème, I et J sont des intervalles de $\mathbf{Z}$, J étant borné, I étant borné inférieurement.

Une donnée $a^q, q = 1, \ldots, u$, intervenant dans les récurrences peut appartenir à une suite simple de la forme $(s_l)_{l \in L}$ ou à une suite double de la forme $(s_{l,l'})_{l \in L, l' \in L'}$ avec L et L' intervalles de $\mathbf{Z}$. Dans ce cas, la donnée a^q est caractérisée par son ou ses indices qui sont définis par une fonction h de i, j et k. La fonction h choisie est une fonction monotone de la variable k. Cela signifie que l'exécution de la récurrence ainsi décrite respecte l'ordre d'indiçage des données utilisées pour faciliter l'accès mémoire (*cf.* chapitre 2). La fonction h n'est pas forcément unique. Il peut y avoir plusieurs énoncés associés à un problème systolisable (voir les exemples traités ci-dessous).

Le résultat du problème est une suite double $(r_{ij})_{i \in I, j \in J}$ où r_{ij} peut être défini de deux façons :
— soit comme étant le résultat de la récurrence, $r_{ij} = y_{ij}^b$,
— soit par $r_{ij} = g(y_{ij}^b, a^1, \ldots, a^u)$ où g est une fonction donnée.

REMARQUE. — Quand le problème s'exprime dans $\mathbf{R}^2$, le système d'équations est défini par un seul indice : i. Le résultat est alors une suite simple $(r_i)_{i \in I}$.

Les exemples suivants illustrent ces différents cas.

EXEMPLE 1. — Considérons le problème du produit de convolution décrit au chapitre 2. Son résultat est une suite simple $(y_i)_{i=0,\ldots,n-m}$. Chaque y_i est le résultat de la récurrence suivante :

$$\begin{cases} y_i^k = y_i^{k-1} + w_l \times x_{i+l} & 0 \leq k \leq m, \quad i = 0, \ldots, n-m \\ y_i^{-1} = 0 \end{cases}$$

où $l = k$ ou $l = m - k$.

EXEMPLE 2. — Considérons le problème de la résolution d'un système triangulaire inférieur de la forme $AX = B$ avec B de dimension n, A de dimension (n, n) telle que $\forall i \in \{1 \ldots n\}, \quad a_{ii} \neq 0$. Le résultat du problème est une suite simple $(x_i)_{i=1,\ldots,n}$. Un résultat $x_i, i = 1, \ldots, n$, est défini en utilisant la récurrence suivante :

$$\begin{cases} s_i^k = s_i^{k-1} + a_{il} \times x_l & 1 \leq k \leq i-1, \quad i = 1, \ldots, n \\ s_i^0 = 0 \end{cases}$$

où $l = k$ ou $l = i - k$, par $x_i = g(s_i^{i-1}, b_i, a_{ii}) = (b_i - s_i^{i-1})/a_{ii}$.

EXEMPLE 3. — Considérons le produit matriciel $C = A \times B$ décrit au chapitre 2 avec $A(m, n)$, $B(n, p)$ et $C(m, p)$. Le résultat est une suite double $(c_{ij})_{i=1,\ldots,m, j=1,\ldots,p}$ où chaque c_{ij} est le résultat de la récurrence à deux indices :

$$\begin{cases} c_{ij}^k = c_{ij}^{k-1} + a_{il} \times b_{lj} & 1 \leq k \leq n, \quad i = 1, \ldots, m \quad , \quad j = 1, \ldots, p \\ c_{ij}^0 = 0 \end{cases}$$

où $l = k$ ou $l = n - k + 1$.

REMARQUE. — Dans les trois exemples, il y a au moins deux énoncés possibles. Ainsi, pour l'exemple 1, l'un correspond à $l = k$ et l'autre à $l = m - k$. Ces deux énoncés correspondent aux deux récurrences respectant l'ordre des indices, soit croissant, soit décroissant. Cela est possible en raison de la commutativité de l'addition dans $\mathbf{R}$.

I.2 Méthode de décomposition du problème. — La méthode de décomposition en modules que nous proposons ici a pour but d'aider à

la détermination d'un énoncé d'un problème. Le problème sera considéré comme systolisable si l'énoncé obtenu est un système d'équations récurrentes de la forme décrite ci-dessus (au paragraphe I.1).

C'est une méthode de type déductive (*cf.* [PAI 79]) qui est basée sur une recherche progressive de l'énoncé à partir du résultat à obtenir. Initialement, le problème est considéré comme un module caractérisé par le résultat à calculer. Une première étape de décomposition consiste à décomposer le résultat associé à ce module en un ensemble de résultats intermédiaires. A chaque résultat intermédiaire est associé un module. La décomposition de chacun de ces modules se poursuit jusqu'à ce qu'aucun résultat intermédiaire nouveau n'apparaisse. On obtient ainsi un ensemble de modules imbriqués, chaque module définissant les opérations permettant le calcul du résultat associé.

Considérons le résultat d'un module à une étape quelconque de la décomposition. Son type peut avoir deux formes conduisant à deux décompositions différentes :

— le résultat est un n-uplet d'objets de même type. Dans ce cas, le module considéré est décomposé *vectoriellement* en n modules. Chacun des n modules est lié à un des n objets composants le type initial.

  Par exemple, un vecteur de dimension n peut être considéré comme le produit cartésien de ses composantes. Le module associé à ce vecteur peut donc être décomposé en n modules. Le résultat associé à chacun de ces modules est une composante du vecteur.

— le résultat est défini par une fonction de deux formes possibles,
  soit $f(x_1, \ldots, x_m)$ où f est une fonction et où les m arguments x_i, de type identique, sont définis par une même fonction dont les paramètres sont des données du problème,
  soit de la forme $g(f(x_1, \ldots, x_m), a^1, \ldots, a^u)$ où la fonction f est définie comme ci-dessus et où $a^1, \ldots, a^u$ sont des données du problème.

  Le module considéré est alors décomposé *fonctionnellement*, dans le premier cas en m modules $M_i, i = 1, \ldots, m$ dont le résultat associé est x_i. Dans le deuxième cas, il est décomposé en $m + 1$ modules, m modules calculant les x_i et un module calculant la fonction g.

Lorsque le résultat n'est pas d'une des deux formes décrites ci-dessus, on ne peut appliquer, ni décomposition vectorielle, ni décomposition fonctionnelle. Le problème est alors considéré comme n'étant pas systolisable.

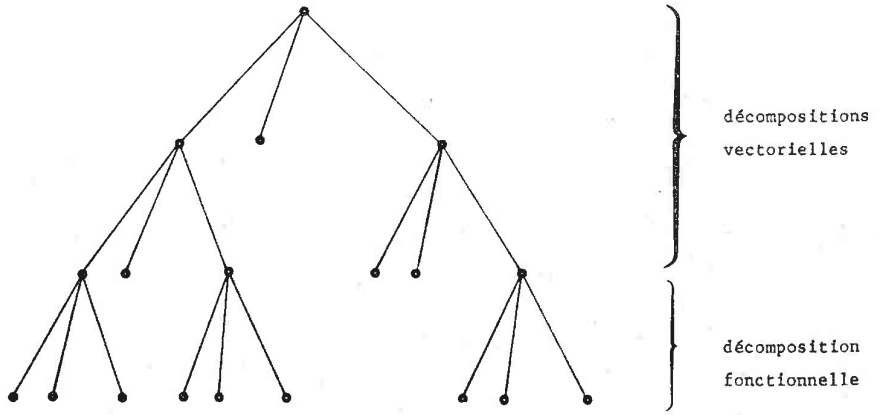


FIGURE 4.1. Arbre des imbrications de modules

Un problème systolisable décomposé selon le processus décrit ci-dessus peut être représenté par un arbre où apparaissent les différents niveaux de modules (*cf.* figure 4.1).

Un module décomposé fonctionellement est donc caractérisé par son (ou ses) indices d'imbrication dans les différents niveaux de modules décomposés vectoriellement. En pratique, les problèmes systolisables sont décomposés en un ou deux niveaux de modules décomposés vectoriellement. Cela correspond aux réseaux linéaires (un niveau) ou bidimensionnels (deux niveaux).

Notons que seule la première décomposition vectorielle peut éventuellement conduire à une infinité de modules de niveau inférieur, par exemple si le résultat associé au module initial est une suite infinie. Cela garantit la terminaison des calculs élémentaires et correspond au fait que l'intervalle I qui définit le premier indice du système d'équations récurrentes peut n'être borné qu'inférieurement.

La détermination de l'énoncé du problème sous forme d'équations récurrentes se fait de la manière suivante.

Considérons les modules décomposés fonctionnellement.

— le ou les indices d'imbrications vectorielles définissent le ou les indices dont dépend le système d'équations.

— la forme de la récurrence est donnée par la fonction f associée à la décomposition fonctionnelle.

Le passage de la fonction f à une forme récurrence équivalente est un processus complexe. C'est le problème rencontré dans le cas général de la construction de programme à partir d'une spécification. Il existe des méthodes permettant, pour certaines formes de f, de déterminer la forme récurrente équivalente. Nous ne détaillons pas ici ces aspects qui nécessitent à eux seuls une étude approfondie.

Notons simplement que dans le cas simple, que l'on rencontre dans les exemples cités ici, où la fonction f est soit l'addition, soit la multiplication dans $\mathbf{R}$, on obtient aisément une forme récurrente. Ces deux opérations étant commutatives, deux énoncés de récurrence sont possibles :

$$\begin{cases} y^k = f(y^{k-1}, x_k) & 1 \leq k \leq m \\ y^0 = e \end{cases}$$

ou

$$\begin{cases} y^k = f(y^{k-1}, x_{m-k+1}) & 1 \leq k \leq m \\ y^0 = e \end{cases}$$

où e est l'élément neutre de f ($e = 0$ pour l'addition, $e = 1$ pour la multiplication).

— lorsque le résultat est défini par $f(x_1, \ldots, x_m)$, l'énoncé du problème est entièrement défini : les résultats élémentaires à calculer sont les résultats de la récurrence ci-dessus.

— lorsque le résultat est défini par $g(f(x_1, \ldots, x_m), a^1, \ldots, a^p)$, les résultats élémentaires à calculer sont définis par $g(s, a^1, \ldots, a^p)$ où s est le résultat de la récurrence ci-dessus.

EXEMPLE. — Considérons le problème du produit matriciel $C = A \times B$ avec A de dimension (m, n), B de dimension (n, p) et C de dimension (m, p). Appliquons la méthode de décomposition proposée.

Première étape de décomposition :

Le résultat du problème est la matrice C, associée au module initial M. Ce résultat est le p-uplet de p objets de même type : les vecteurs colonnes $C^1, \ldots, C^p$ de la matrice C.

Le module M est donc décomposé vectoriellement en une famille de p modules $(M_i)_{i=1,\ldots,p}$. Le résultat associé à chaque module M_i est le vecteur colonne C^i dont le calcul est défini par $C^i = A \times B^i$.

Deuxième étape de décomposition :

Considérons un module quelconque M_i issu de la première décomposition. Son résultat associé est un vecteur C^i de dimension m. C'est un m-uplet de m objets de même type : les composantes $c^i_1, \ldots, c^i_m$. Chaque module M_i, i fixé, est donc décomposé vectoriellement en une famille de m modules $(M_{ij})_{j=1,\ldots,m}$. Le résultat associé à chaque module M_{ij} est la composante c_j du vecteur C^i, c'est-à-dire la composante c_{ij} de la matrice C. Il est défini par

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$$

Troisième étape de décomposition :

Considérons un module $M_{ij}, i = 1, \ldots, p, j = 1, \ldots, m$ issu de la deuxième étape. Son résultat est un produit scalaire défini par une fonction f d'addition sur $\mathbf{R}$:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$$

Chaque module M_{ij} est donc décomposé fonctionnellement en n modules identiques $(M^k_{ij})_{k=1,\ldots,n}$. Le résultat associé au module M^k_{ij} est le produit $a_{ik} \times b_{kj}$.

Déterminons l'énoncé de ce problème sous forme de système d'équations récurrentes. Pour cela, considérons les modules issus de la décomposition fonctionnelle : $M^k_{ij}, i = 1, \ldots, p, j = 1, \ldots, m, k = 1, \ldots, n$. Ils sont caractérisés par deux indices d'imbrication vectorielle, i et j. Le système d'équations dépend donc de deux indices, $i = 1, \ldots, p$ et $j = 1, \ldots, m$.

La fonction f associée à la décomposition fonctionnelle est l'addition dans $\mathbf{R}$:

$$\sum_{k=1}^{n} a_{ik} \times b_{kj}$$

On a donc deux énoncés de récurrence possibles :

$$\begin{cases} c^k_{ij} = c^{k-1}_{ij} + a_{ik} \times b_{kj} & 1 \leq k \leq n \\ c^0_{ij} = 0 \end{cases}$$

et

$$\begin{cases} c^k_{ij} = c^{k-1}_{ij} + a_{i,n-k+1} \times b_{n-k+1,j} & 1 \leq k \leq n \\ c^0_{ij} = 0 \end{cases}$$

Un résultat élémentaire du problème est un résultat de la récurrence : $c_{ij} = c_{ij}^n, i = 1, \ldots, m, j = 1, \ldots, p$.

II. Spécification et représentation d'un problème systolisable

Comme nous l'avons vu au paragraphe précédent, l'énoncé d'un problème systolisable défini dans $\mathbf{R}^3$ est donné par un système d'équations récurrentes de la forme :

$$(1)\begin{cases} y_{ij}^k = f(y_{ij}^{k-1}, a^1, \ldots, a^u), & a < k \leq b, \quad i \in I, \quad j \in J \\ y_{ij}^a = v, & v \in \mathbf{R} \end{cases}$$

Le résultat du problème est une suite double $(r_{ij})_{i\in I, j\in J}$.
Un résultat élémentaire $r_{ij}, i \in I, j \in J$ peut être défini
— par $r_{ij} = y_{ij}^b$
— ou par $r_{ij} = g(y_{ij}^b, a^1, \ldots, a^u)$.

Nous déterminons, à partir de cet énoncé, une spécification systolique du problème. Cette spécification est définie par
— le domaine associé au problème définissant l'ensemble de ces calculs élémentaires,
— la fonction de calcul réalisée et la liste des données utilisées par chaque calcul élémentaire.

Nous définissons ensuite de manière formelle ce qu'est une représentation du problème.

II.1 Le domaine associé au problème. — Le domaine D associé à un problème donné est l'ensemble des points de coordonnées entières correspondant aux calculs élémentaires. Il est défini à partir du système d'équations définissant le problème.

Dans la pratique, D est un sous-ensemble de $\mathbf{Z}^r$ avec $r = 2$ ou 3. Dans la suite, on prend $r = 3$. Les simplifications pour $r = 2$ sont évidentes.

DÉFINITION 1. — Soit un problème systolisable défini dans $\mathbf{R}^3$, dont l'énoncé est donné par le système d'équations récurrentes de la forme (1).

Le domaine D associé au problème est la réunion de deux sous-domaines D_1 et D_2.
Le *sous-domaine* D_1 est l'ensemble des valeurs des indices définissant le système d'équations récurrentes. Il est défini par :

$$D_1 = \{(i, j, k) \in \mathbf{Z}^3 \mid i \in I, \quad j \in J, \quad a < k \leq b\}$$

Le *sous-domaine* D_2 est défini de la façon suivante :
(i) Si le résultat du problème est

$$(r_{ij})_{i\in I, j\in J} \quad tel \quad que \quad r_{ij} = y_{ij}^b,$$

alors $D_2 = \emptyset$.
(ii) Si le résultat du problème est

$$(r_{ij})_{i\in I, j\in J} \quad tel \quad que \quad r_{ij} = g(y_{ij}^b, a^1, \ldots, a^u),$$

alors $D_2 = \{(i, j, k) \in \mathbf{Z}^3 \mid i \in I, \quad j \in J, \quad k = b + 1\}$.

REMARQUE. — Lorsque le problème peut être défini par plusieurs énoncés de récurrence, le domaine est le même dans tous les cas. En effet, les calculs élémentaires sont les mêmes, seul leur ordre est modifié.

Considérons à nouveau les trois problèmes évoqués précédemment.

EXEMPLE 1. — Un énoncé du produit de convolution est donné par la récurrence :

$$\begin{cases} y_i^k = y_i^{k-1} + w_k \times x_{i+k}, & 0 \leq k \leq m, \quad i = 0, \ldots, n - m \\ y_i^{-1} = 0 \end{cases}$$

Le résultat est une suite $(y_i)_{i=0,\ldots,n-m}$ définie par $y_i = y_i^m$.
Le sous-domaine D_1 est donc défini par

$$D_1 = \{(i, k) \in \mathbf{Z}^2 \mid 0 \leq i \leq n - m, \quad 0 \leq k \leq m\}$$

Le sous-domaine D_2 étant vide car un résultat élémentaire y_i est égal à un résultat de récurrence, le domaine est $D = D_1$.

EXEMPLE 2. — Un énoncé de la résolution d'un système triangulaire inférieur $AX = B$ est :

$$\begin{cases} s_i^k = s_i^{k-1} + a_{ik} \times x_k, & 1 \leq k \leq i-1, \quad i = 1, \ldots, n \\ s_i^0 = 0 \end{cases}$$

Le résultat est une suite $(x_i)_{i=1,\ldots,n}$ définie par $x_i = g(s_i^{i-1}, b_i, a_{ii}) = (b_i - s_i^{i-1})/a_{ii}$.

Le sous-domaine D_1 est donc défini par

$$D_1 = \{(i,k) \in \mathbf{Z}^2 \quad | \quad 1 \leq i \leq n, \quad 1 \leq k \leq i-1\}$$

Le sous-domaine D_2 est défini par

$$D_2 = \{(i,k) \in \mathbf{Z}^2 \quad | \quad 1 \leq i \leq n, \quad k = i\}$$

Le domaine D est alors

$$D = D_1 \bigcup D_2 = \{(i,k) \in \mathbf{Z}^2 \quad | \quad 1 \leq i \leq n, \quad 1 \leq k \leq i\}$$

EXEMPLE 3. — Un énoncé du produit matriciel $C = A \times B$ est :

$$\begin{cases} c_{ij}^k = c_{ij}^{k-1} + a_{ik} \times b_{kj}, & 1 \leq k \leq n, \quad i = 1, \ldots, m, \quad j = 1, \ldots, p \\ c_{ij}^0 = 0 \end{cases}$$

Le résultat est la suite $(c_{ij})_{i=1,\ldots,m,j=1,\ldots,p}$ définie par $c_{ij} = c_{ij}^n$.
On a alors $D_2 = \emptyset$, d'où :

$$D = D_1 = \{(i,j,k) \in \mathbf{Z}^3 \quad | \quad 1 \leq i \leq m, \quad 1 \leq j \leq p, \quad 1 \leq k \leq n\}$$

Rappelons que seule la première composante d'un point de D peut ne pas être bornée. En effet, comme nous l'avons indiqué au pararagraphe I.1, seul l'intervalle I définissant le premier indice du système d'équations récurrentes, peut ne pas être borné supérieurement.

Tout point du domaine $D \in \mathbf{Z}^3$ associé au problème s'exprime de manière unique sur la base $(\vec{b}_i, \vec{b}_j, \vec{b}_k)$ où $\vec{b}_i = (1,0,0), \vec{b}_j = (0,1,0), \vec{b}_k = (0,0,1)$. Nous appelons cette base la *base canonique* du problème.

II.2. Spécification systolique du problème. — Soit un problème systolisable défini dans $\mathbf{R}^3$, à partir de p suites de données, par un énoncé de la forme (1).

Le domaine D associé au problème définit les coordonnées des calculs élémentaires du problème. Ces calculs sont définis par la fonction réalisée et les arguments de cette fonction, c'est-à-dire les données utilisées.

Appelons famille de ressources une suite simple ou double de données intervenant dans la définition du problème. La suite de résultats est considérée comme une famille. Une famille peut être réduite à un élément (suite simple à un seul élément).

A chaque point de coordonnées (i,j,k) du domaine D sont associés :
(1) la fonction de récurrence f, si ce point est dans D_1, ou la fonction g, si ce point est dans D_2,
(2) pour chacune des p familles de ressources du problème, l'indice de l'élément de cette suite utilisé en ce point. Pour la q-ième suite ($q = 1 \ldots p$), il est défini par une fonction h_q déterminée par :

— (i) si la q-ième famille de ressources est une suite simple de la forme $(s_l)_{l \in L}$:

$$\begin{array}{rccc} h_q : & \mathbf{Z}^3 & \rightarrow & \mathbf{Z} \\ & (i,j,k) & \rightarrow & h(i,j,k) = l \end{array}$$

— (ii) si la q-ième famille de ressources est une suite double de la forme $(s_{ll'})_{l \in L, l' \in L'}$:

$$\begin{array}{rccc} h_q : & \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k) & \rightarrow & h(i,j,k) = (l,l') \end{array}$$

— (iii) si la q-ième famille n'est pas utilisée aux points (i,j,k) considérés :

$$\begin{array}{rccc} h_q : & \mathbf{Z}^3 & \rightarrow & \{\Lambda\} \\ & (i,j,k) & \rightarrow & h(i,j,k) = \Lambda \end{array}$$

REMARQUE 1. — Lorsque le problème est linéaire, le profil des fonctions h est $\mathbf{Z}^2 \rightarrow A$ avec $A = \mathbf{Z}, \mathbf{Z}^2 ou \{\Lambda\}$ (l'indice j n'apparait pas).

REMARQUE 2. — La fonction h associée à la suite de résultats du problème est définie par,

$$\begin{array}{rccc} h : & \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k) & \rightarrow & h(i,j,k) = (i,j) \end{array}$$

Cela signifie qu'un résultat r_{ij} est utilisé par tous les points d'indice i, j. Seul son indice de récurrence k diffère d'un point à un autre.

Nous pouvons maintenant définir ce que nous appelons la spécification systolique du problème, c'est-à-dire l'ensemble des informations utilisées par la méthode que nous proposons pour obtenir des algorithmes systoliques solution du problème. Notons que l'emploi du terme spécification est un abus de langage, la spécification systolique étant liée à la méthode de résolution proposée.

DÉFINITION 2. — La *spécification systolique* d'un problème défini dans $\mathbf{R}^3$, à partir de p familles de données est caractérisée par :
— le domaine $D \subset \mathbf{Z}^3$ associé au problème. Il définit les coordonnées des calculs élémentaires dans la base canonique $(\vec{b}_i, \vec{b}_j, \vec{b}_k)$ du problème,
— pour chaque point du domaine, la fonction associée à ce point et la liste des données utilisées sous la forme $\{h_1, \ldots, h_p\}$ où les fonctions $h_i, i = 1, \ldots, p$, sont décrites ci-dessus.

Considérons à nouveau les trois exemples exposés précédemment.

EXEMPLE 1. — Le produit de convolution est défini par trois familles de données : $(w_k)_{k=0,\ldots,m}$, $(x_i)_{i=0,\ldots,n}$ et $(y_i)_{i=0,\ldots,n-m}$. Sa spécification systolique est donnée par :
— le domaine $D = D_1 = \{(i,k) \in \mathbf{Z}^2 \mid 0 \leq i \leq n-m, \quad 0 \leq k \leq m\}$ (*cf.* par.II.1).
Comme $D_2 = \emptyset$, les informations associées à tout point de D sont les mêmes. Ainsi :
— la fonction associée en un point de D est définie par

$$\begin{array}{rccc} f: & \mathbf{R}^3 & \rightarrow & \mathbf{R} \\ & (y,w,x) & \rightarrow & f(y,w,x) = y + w \times x \end{array}$$

— la liste des données utilisées en un point de D est définie par l'ensemble $\{h_y, h_w, h_x\}$ avec

$$\begin{array}{rccc} h_y: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,k) & \rightarrow & h_y(i,k) = i \end{array}$$

$$\begin{array}{rccc} h_w: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,k) & \rightarrow & h_w(i,k) = k \end{array}$$

$$\begin{array}{rccc} h_x: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,k) & \rightarrow & h_x(i,k) = i + k \end{array}$$

EXEMPLE 2. — La résolution d'un système triangulaire inférieur est défini par trois familles de données $(a_{ij})_{i=1,\ldots,n, j=1,\ldots,i}$, $(b_i)_{i=1,\ldots,n}$ et $(x_i)_{i=1,\ldots,n}$. Sa spécification systolique est donnée par :
— le domaine $D = D_1 \bigcup D_2 = \{(i,k) \in \mathbf{Z}^2 \mid 1 \leq i \leq n, 1 \leq k \leq i\}$

Pour tout point de $D_1 = \{(i,k) \in \mathbf{Z}^2 \mid 1 \leq i \leq n, 1 \leq k < i\}$:
— la fonction associée définie par :

$$\begin{array}{rccc} f: & \mathbf{R}^3 & \rightarrow & \mathbf{R} \\ & (s,a,x) & \rightarrow & f(s,a,x) = s + a \times x \end{array}$$

— la liste des données utilisées définie par $\{h_x, h_a, h_b\}$ avec

$$\begin{array}{rccc} h_x: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,k) & \rightarrow & h_x(i,k) = k \end{array}$$

$$\begin{array}{rccc} h_a: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z}^2 \\ & (i,k) & \rightarrow & h_a(i,k) = (i,k) \end{array}$$

$$\begin{array}{rccc} h_b: & \mathbf{Z}^2 & \rightarrow & \{\Lambda\} \\ & (i,k) & \rightarrow & h_b(i,k) = \Lambda \end{array}$$

Notons qu'une donnée de la suite des résultats $(x_i)_{i=1,\ldots,n}$ est utilisée comme argument de la fonction f pour calculer son résultat s. Il s'agit d'un cas particulier de problème où les résultats à calculer servent aussi de données. Dans ce cas, la fonction h_x décrit quel élément de la suite est utilisé comme donnée au point considéré.

Pour tout point de $D_2 = \{(i,i) \in \mathbf{Z}^2 \mid 1 \leq i \leq n\}$:
— la fonction associée définie par :

$$\begin{array}{rccc} g: & \mathbf{R}^3 & \rightarrow & \mathbf{R} \\ & (x,a,b) & \rightarrow & g(x,a,b) = (b-x)/a \end{array}$$

— la liste des données utilisées définie par $\{h_x, h_a, h_b\}$ avec

$$\begin{array}{rccc} h_x: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,i) & \rightarrow & h_x(i,i) = i \end{array}$$

$$\begin{array}{rccc} h_a: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z}^2 \\ & (i,i) & \rightarrow & h_a(i,i) = (i,i) \end{array}$$

$$\begin{array}{rccc} h_b: & \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,i) & \rightarrow & h_b(i,i) = i \end{array}$$

EXEMPLE 3. — Le produit matriciel est défini par trois familles de données $(a_{ij})_{i=1,\ldots,m,j=1,\ldots,n}$, $(b_{ij})_{i=1,\ldots,n,j=1,\ldots,p}$ et $(c_{ij})_{i=1,\ldots,m,j=1,\ldots,p}$. Sa spécification systolique est définie par :

— le domaine $D = D_1 = \{(i,j,k) \in \mathbf{Z}^3 \mid 1 \leq i \leq m, 1 \leq j \leq p, 1 \leq k \leq n\}$

— la fonction associée en un point de D, définie par :

$$\begin{array}{rccc} f: & \mathbf{R}^3 & \rightarrow & \mathbf{R} \\ & (c,a,b) & \rightarrow & f(c,a,b) = c + a \times b \end{array}$$

— la liste des données utilisées en un point de D définie par $\{h_c, h_a, h_b\}$ avec

$$\begin{array}{rccc} h_c: & \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k) & \rightarrow & h_c(i,j,k) = (i,j) \\ h_a: & \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k) & \rightarrow & h_a(i,j,k) = (i,k) \\ h_b: & \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k) & \rightarrow & h_b(i,j,k) = (k,j) \end{array}$$

II.3. Les vecteurs générateurs. — Comme nous l'avons vu au chapitre 3, dans un problème systolisable, une donnée peut être utilisée dans plusieurs calculs élémentaires, c'est-à-dire en plusieurs points du domaine. Un algorithme systolique fournit un certain ordonnancement de ces calculs élémentaires. De cet ordonnancement dépend la façon d'utiliser la donnée dans le réseau (diffusion, donnée constante d'une cellule, circulation sur un flot). La notion de vecteur générateur est introduite pour définir les points du domaine utilisant la même donnée et faciliter ainsi la détermination des réseaux correspondants.

DÉFINITION 3. — Soit un problème systolisable défini dans $\mathbf{R}^3$, à partir de p familles de données. Soit d une famille de données dont la fonction associée h_d est définie dans la spécification systolique du problème.

On appelle *vecteur générateur* associé à la famille d et on le note Φ_d, un vecteur de $\mathbf{Z}^3$ de coordonnées $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ dans la base canonique (B.C.) du problème, tel que :

— pour un point quelconque (i,j,k) du domaine D on a

$$h_d(i,j,k) = h_d(i + \varphi_i, j + \varphi_j, k + \varphi_k)$$

— PGCD $(\varphi_i, \varphi_j, \varphi_k) = \pm 1$

Cette définition du vecteur générateur Φ_d traduit le fait que les points (i,j,k) et $(i+\varphi_i, j+\varphi_j, k+\varphi_k)$ du domaine utilisent une même occurrence de la famille de données d.

Notons que le vecteur Φ_d n'est pas unique. Le fait de choisir Φ_d tel que ses coordonnées soient premières entre elles (PGCD $(\varphi_i, \varphi_j, \varphi_k) = \pm 1$) permet de limiter les choix possibles pour Φ_d et d'obtenir, à partir d'un point quelconque (i,j,k) de D, tous les points $(i + n \times \varphi_i, j + n \times \varphi_j, k + n \times \varphi_k), n \in \mathbf{Z}$, du domaine D utilisant la même occurrence de la donnée d.

REMARQUES .

— Pour un problème linéaire (dans $\mathbf{R}^2$), les vecteurs générateurs sont des vecteurs de $\mathbf{Z}^2$ de la forme $\Phi_d = (\varphi_i, \varphi_k)_{B.C.}$.

— pour une famille de données d dont les différentes occurrences ne sont utilisées qu'une fois, c'est-à-dire quand on a $h_d(i,j,k) = \Lambda$, le vecteur générateur associé Φ_d est nul.

— considérons la famille de résultats r du problème. On a $h_r(i,j,k) = (i,j)$. Soit $\Phi_r = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ le vecteur générateur associé à cette famille. On a

$$h_r(i,j,k) = h_r(i + \varphi_i, j + \varphi_j, k + \varphi_k) \Leftrightarrow (i,j) = (i + \varphi_i, j + \varphi_j)$$

Donc $\varphi_i = \varphi_j = 0$.

La composante φ_k peut être choisie égale à ± 1. On adopte la convention que $\varphi_k = 1$. Le vecteur générateur associé à la famille de résultats est donc $\Phi_r = (0,0,1)_{B.C.}$.

EXEMPLE. — Pour le produit de convolution, on a :

— (1) $h_w(i,k) = k$

Le vecteur générateur $\Phi_w = (\varphi_i, \varphi_k)_{B.C.}$ vérifie

$$h_w(i,k) = h_w(i + \varphi_i, k + \varphi_k) \Leftrightarrow k = k + \varphi_k \Leftrightarrow \varphi_k = 0$$

Comme de plus, la définition 3 impose que PGCD $(\varphi_i, \varphi_k) = \pm 1$, on a $\varphi_i = \pm 1$. Le vecteur générateur Φ_w peut donc être $(1,0)$ ou $(-1,0)$ exprimé dans la base canonique.

— (2) $h_x(i,k) = i + k$

Le vecteur générateur $\Phi_x = (\varphi_i, \varphi_k)_{B.C.}$ vérifie

$$h_x(i,k) = h_x(i+\varphi_i, k+\varphi_k) \Leftrightarrow i+k = i+k+\varphi_i+\varphi_k$$

$$\Leftrightarrow \varphi_i + \varphi_k = 0 \Leftrightarrow \varphi_i = -\varphi_k$$

De plus, on a PGCD $(\varphi_i, \varphi_k) = \pm 1$. On peut donc avoir $\Phi_x = (1,-1)$ ou $\Phi_x = (-1,1)$ dans la base canonique.

— (3) En raison de la remarque ci-dessus , on a $\Phi_y = (0,1)_{B.C.}$.

II.4. Représentation d'un problème. — A un problème donné défini dans $\mathbf{R}^3$, on associe un ensemble de représentations. Chaque représentation définit un ordonnancement des calculs élémentaires. Comme nous l'avons indiqué au chapitre 3, cette relation ordre temporel entre les calculs nécessite l'introduction d'un axe des temps. Il est choisi parallèle à l'axe de récurrence car cette relation est un ordre total sur les calculs de récurrence associés à un même résultat élémentaire.

On appelle alors *espace-temps*, l'espace $\mathbf{R}^3$ muni de la base orthonormée $(\vec{i}, \vec{j}, \vec{t})$ où le dernier vecteur de base $\vec{t}$ représente l'axe des temps.

Dans la méthode proposée, les seuls ordonnancements autorisés sont tels que les vecteurs de la base canonique sont conservés dans l'espace-temps. D'où la définition suivante :

DÉFINITION 4. — Une *représentation* d'un problème dans l'espace-temps est donnée par

— la matrice de passage P de la base canonique $(B.C.)$ du domaine à la base de l'espace-temps $(E.T.)$.

— le vecteur de translation $V = \overrightarrow{O'O}_{(E.T.)}$ où O est l'origine du repère associé à la base canonique et O' l'origine du repère de l'espace-temps.

Les coordonnées d'un point dans l'espace-temps s'expriment alors en fonction de ses coordonnées dans la base canonique par :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{E.T.} = P \begin{pmatrix} i \\ j \\ k \end{pmatrix}_{B.C.} + V$$

Inversement, les coordonnées d'un point dans la base canonique sont données par :

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix}_{B.C.} = P^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{E.T.} - P^{-1} V$$

Nous appelons *représentation initiale* d'un problème que nous notons R_0, la représentation pour laquelle il y a coïncidence entre la base canonique et la base de l'espace-temps, c'est-à-dire pour laquelle $P = I$, I étant la matrice identité, et V le vecteur nul.

Le schéma de la représentation initiale du produit de convolution est visualisé figure 3-3.

Le problème peut imposer des conditions de disponibilité des données. Ces contraintes, liées au problème, peuvent être de deux types :

— restrictions sur l'accès aux données. Les seules prises en compte sont celles imposant des délais sur l'accès aux éléments d'une famille de données. Par exemple, la suite de données $(x_i)_{i\geq 0}$ est disponible élément par élément aux instants $t, t+\delta, t+2\delta$, etc, avec $\delta > 0$. Cette contrainte doit s'exprimer par une information supplémentaire ajoutée à la spécification systolique du problème.

— un résultat élémentaire du problème peut être utilisé comme donnée dans le calcul d'un autre résultat élémentaire (*cf.* l'exemple de la résolution d'un système triangulaire). Une telle contrainte s'exprime par la fonction h associée au résultat. Elle n'est définie par $h(i,j,k) = (i,j)$ que dans le cas où le résultat n'est pas utilisé comme donnée. Dans l'autre cas, elle définit comment le résultat est utilisé comme donnée.

Lorsque de telles contraintes existent, la représentation initiale du problème, ainsi que d'autres représentations, peut ne pas les vérifier. Dans ce cas, les réseaux systoliques associés à ces représentations ne sont pas solution du problème posé.

On appelle alors *représentation valide*, une représentation du problème pour laquelle toutes les contraintes d'accès aux données liées au problème sont vérifiées.

Seules les représentations valides conduisent à des réseaux systoliques solution du problème posé. Lorsqu'il n'y a pas de contraintes d'accès aux

données, toute représentation du problème est valide. C'est le cas pour les exemples du produit de convolution est du produit matriciel.

III. Transformation appliquée à une représentation

Comme nous l'avons exposé précédemment, nous souhaitons obtenir un ensemble de représentations du problème, chaque représentation étant caractérisée par un certain ordonnancement des calculs élémentaires. Notre objectif est d'obtenir des réseaux différents associés à chacunes de ces représentations.

Ces représentations sont obtenues à partir de la représentation initiale, notée R_0, définie au paragraphe II.4 en appliquant une suite de transformations. Une transformation appliquée à une représentation R permet de définir une nouvelle représentation, donc un ordre différent des calculs. Les conditions méthodologiques d'application d'une transformation sont étudiées ensuite.

Nous nous situons dans $\mathbf{R}^3$ avec $(O', \vec{i}, \vec{j}, \vec{t})$ comme repère de l'espace-temps. Cette restriction à $\mathbf{R}^3$ correspond aux cas pratiques de problèmes résolus par des réseaux linéaires ($\mathbf{R}^2$) ou bidimensionnels ($\mathbf{R}^3$). Les simplifications pour $\mathbf{R}^2$ sont évidentes (suppression de la composante sur l'axe $(O'\vec{j})$).

DÉFINITION 5. — Soit R une représentation définie par P et V dans l'espace-temps. Une *transformation* appliquée à la représentation R, notée T_R, est une application R-affine définie par :

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i, j, \alpha i + \beta j + \gamma t + \delta)_{E.T.} \end{array}$$

où $\alpha, \beta, \gamma, \delta$ sont des constantes entières.

Nous employons le terme de R-affine pour indiquer que la transformation T_R est une application affine paramétrée par la représentation R, et que l'image de R par T_R doit être une représentation.

PROPRIÉTÉ 1. — *Soit R une représentation définie par*
— *une matrice de passage P définissant la base canonique dans l'espace-temps :*

$$P = \begin{pmatrix} b_i^1 & b_j^1 & b_k^1 \\ b_i^2 & b_j^2 & b_k^2 \\ b_i^3 & b_j^3 & b_k^3 \end{pmatrix}$$

— *un vecteur de translation V définissant l'origine du repère associé à la base canonique dans l'espace-temps :*

$$V = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Son image par une transformation T_R est la représentation, notée $T_R(R)$, définie par
— *la matrice de passage :*

$$P' = \begin{pmatrix} b_i^1 & b_j^1 & b_k^1 \\ b_i^2 & b_j^2 & b_k^2 \\ \alpha b_i^1 + \beta b_i^2 + \gamma b_i^3 & \alpha b_j^1 + \beta b_j^2 + \gamma b_j^3 & \alpha b_k^1 + \beta b_k^2 + \gamma b_k^3 \end{pmatrix}$$

— *le vecteur de translation :*

$$V' = \begin{pmatrix} v_1 \\ v_2 \\ \alpha v_1 + \beta v_2 + \gamma v_3 + \delta \end{pmatrix}$$

Démonstration. — Un point de coordonnées $(i,j,k)_{B.C.}$ a pour coordonnées dans l'espace-temps sur la représentation R, $(i',j',t)_{E.T.}$ définies par :

$$\begin{pmatrix} i' \\ j' \\ t \end{pmatrix}_{E.T.} = P \begin{pmatrix} i \\ j \\ k \end{pmatrix}_{B.C.} + V = \begin{pmatrix} b_i^1 i + b_j^1 j + b_k^1 k + v_1 \\ b_i^2 i + b_j^2 j + b_k^2 k + v_2 \\ b_i^3 i + b_j^3 j + b_k^3 k + v_3 \end{pmatrix}$$

L'image du point $(i',j',t)_{E.T.}$ par une transformation T_R décrite définition 5, est le point $(i',j',t')_{E.T.}$ de la représentation $T_R(R)$ avec :

$$t' = \alpha i' + \beta j' + \gamma t + \delta \Leftrightarrow$$

$$t' = \alpha(b_i^1 i + b_j^1 j + b_k^1 k + v_1) + \beta(b_i^2 i + b_j^2 j + b_k^2 k + v_2) + \gamma(b_i^3 i + b_j^3 j + b_k^3 k + v_3) + \delta \quad (1)$$

Supposons la représentation $T_R(R)$ définie par P' et V'. La transformation T_R ne modifie pas les deux premières composantes des points dans l'espace-temps. Les deux premières lignes de P' et V' sont donc les mêmes que celles de P et V. On a donc :

$$P' = \begin{pmatrix} b_i^1 & b_j^1 & b_k^1 \\ b_i^2 & b_j^2 & b_k^2 \\ b'^3_i & b'^3_j & b'^3_k \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} v_1 \\ v_2 \\ v'_3 \end{pmatrix}$$

Le point $(i', j', t')_{E.T.}$ de la représentation $T_R(R)$ est donc défini par

$$\begin{pmatrix} i' \\ j' \\ t' \end{pmatrix}_{E.T.} = P' \begin{pmatrix} i \\ j \\ k \end{pmatrix}_{B.C.} + V'$$

D'où $t' = b'^3_i i + b'^3_j j + b'^3_k k + v'_3 \qquad (2)$.
De (1) et (2) on déduit :

$$\begin{cases} b'^3_i = \alpha b_i^1 + \beta b_i^2 + \gamma b_i^3 \\ b'^3_j = \alpha b_j^1 + \beta b_j^2 + \gamma b_j^3 \\ b'^3_k = \alpha b_k^1 + \beta b_k^2 + \gamma b_k^3 \\ v'_3 = \alpha v_1 + \beta v_2 + \gamma v_3 + \delta \end{cases}$$

d'où le résultat énoncé ci-dessus.

La transformation affine T_R est définie par une matrice M_{T_R} et un vecteur V_{T_R} définis par :

$$M_{T_R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & \gamma \end{pmatrix} \quad \text{et} \quad V_{T_R} = \begin{pmatrix} 0 \\ 0 \\ \delta \end{pmatrix}$$

L'image d'un point $(i, j, t)_{E.T.}$ du domaine par cette transformation est un point défini par :

$$\begin{pmatrix} i' \\ j' \\ t' \end{pmatrix} = M_{T_R} \times \begin{pmatrix} i \\ j \\ t \end{pmatrix} + V_{T_R}$$

La représentation $T_R(R)$ image de la représentation R par la transformation T_R est définie par P' et V' avec

$$P' = M_{T_R} \times P$$
$$V' = M_{T_R} \times V + V_{T_R}$$

On constate donc qu'une transformation n'affecte pas les deux premières composantes des vecteurs de P et du vecteur V. Or initialement, la matrice P est la matrice identité et le vecteur de translation V est nul. Cela signifie que toute représentation est définie par P et V avec $b_i^1 = b_j^2 = 1 \quad b_i^2 = b_j^1 = b_k^1 = b_k^2 = v_1 = v_2 = 0$. On obtient donc les simplifications suivantes de la propriété 1 :

THÉORÈME 1. — *Une représentation quelconque R d'un problème est définie dans l'espace-temps par :*
— *une matrice de passage P de la base canonique à l'espace-temps de la forme*

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix}$$

— *un vecteur de translation $V = (0, 0, v)^t$*
Son image par une transformation T_R décrite définition 5 est une représentation $T_R(R)$ définie par :
— *la matrice de passage*

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha + \gamma b_1 & \beta + \gamma b_2 & \gamma b_3 \end{pmatrix}$$

— *le vecteur de translation $V' = (0, 0, \gamma v + \delta)^t$*
où V^t est le vecteur transposé de V.

PROPRIÉTÉ 2. — *La composition de deux applications R-affines T_1 et T_2 définies par*

$$\begin{array}{rccc} T_1 : & \mathbf{R}^3 & \to & \mathbf{R}^3 \\ & (i, j, t)_{E.T.} & \to & (i, j, \alpha_1 i + \beta_1 j + \gamma_1 k + \delta_1)_{E.T.} \end{array}$$

et

$$\begin{array}{rccc} T_2 : & \mathbf{R}^3 & \to & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \to & (i,j,\alpha_2 i+\beta_2 j+\gamma_2 k+\delta_2)_{E.T.} \end{array}$$

est une application R-affine définie par

$$\begin{array}{rccc} T_2 \circ T_1 : & \mathbf{R}^3 & \to & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \to & (i,j,(\alpha_1\gamma_2+\alpha_2)i+(\beta_1\gamma_2+\beta_2)j \\ & & & +(\gamma_1\gamma_2)t+(\delta_1\gamma_2+\delta_2))_{E.T.} \end{array}$$

Démonstration. — Considérons une représentation R définie par

$$P=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V=\begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

D'après la propriété 1, l'image de la représentation R par la transformation T_1 paramétrée par R est la représentation $R_1 = T_{1_R}(R)$ définie par

$$P_1=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha_1+\gamma_1 b_1 & \beta_1+\gamma_1 b_2 & \gamma_1 b_3 \end{pmatrix} \quad \text{et} V_1=\begin{pmatrix} 0 \\ 0 \\ \gamma_1 v+\delta_1 \end{pmatrix}$$

A la représentation R_1 ainsi obtenue, nous appliquons la transformation T_2 paramétrée par R_1 caractérisée par les constantes $\alpha_2, \beta_2, \gamma_2, \delta_2$ dépendant des composantes de P_1 et V_1. La représentation image de R_1 par cette transformation est la représentation $R_2 = T_{2_{R_1}}(R_1) = (T_2 \circ T_1)_R(R)$ définie par

$$P_2=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha_2+\gamma_2(\alpha_1+\gamma_1 b_1) & \beta_2+\gamma_2(\beta_1+\gamma_1 b_2) & \gamma_2(\gamma_1 b_3) \end{pmatrix}$$

$$=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ (\alpha_1\gamma_2+\alpha_2)+(\gamma_1\gamma_2)b_1 & (\beta_1\gamma_2+\beta_2)+(\gamma_1\gamma_2)b_2 & (\gamma_1\gamma_2)b_3 \end{pmatrix}$$

et

$$V_2=\begin{pmatrix} 0 \\ 0 \\ \gamma_2(\gamma_1 v+\delta_1)+\delta_2 \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ (\gamma_1\gamma_2)v+(\delta_1\gamma_2+\delta_2) \end{pmatrix}$$

d'où l'on déduit la définition de $T_2 \circ T_1$ donnée ci-dessus.

Comme nous l'avons indiqué lors de la présentation intuitive de la méthode (cf. chapitre 3, par. III), les transformations effectives appliquées sont des formes particulières de la définition générale donnée ci-dessus (définition 5). Dans $\mathbf{R}^2$, les droites (D_t) constituées des points de même abscisse temporelle pour une représentation R donnée ont pour image par une transformation des droites parallèles de vecteur directeur $\vec{i} \pm p\vec{t}$ (c'est-à-dire de pente $\pm p$), avec $p = 1$ ou $p = 1/\tau$, τ étant l'abscisse temporelle du vecteur $\vec{b}_k$. De même dans $\mathbf{R}^3$, muni du repère $(O', \vec{i}, \vec{j}, \vec{t})$ de l'espace-temps, les plans (P_t) constitués des points de même abscisse temporelle pour une représentation R donnée, ont pour image par une transformation T_R sur la représentation $T_R(R)$ des plans parallèles à l'un des plans P_i ou P_j, respectivement associés à l'axe $(O'\vec{i})$ ou à l'axe $(O'\vec{j})$ et définis comme suit : un plan P_i est engendré par les vecteurs $\vec{j}$ et $\vec{i} \pm p\vec{t}$, un plan P_j est engendré par les vecteurs $\vec{i}$ et $\vec{j} \pm p\vec{t}$, avec $p = 1$ ou $p = 1/\tau$.

Dans $\mathbf{R}^3$, on distingue donc deux catégories de transformations, celles en référence à l'axe $(O'\vec{i})$ et celles en référence à l'axe $(O'\vec{j})$. De plus, pour les transformations en référence à un axe, par exemple $(O'\vec{i})$, on distingue deux plans P_i, l'un défini par les vecteurs $\vec{j}$ et $\vec{i} + p\vec{t}$ et noté P_{i_c}, l'autre défini par les vecteurs $\vec{j}$ et $\vec{i} - p\vec{t}$ et noté P_{i_d}. Cela conduit à deux types de transformations :

— les transformations, dites $T_c(O'\vec{i})$, pour lesquelles les images des plans (P_t) sont des plans parallèles au plan P_{i_c}. Cela correspond pour $\mathbf{R}^2$ aux droites (D_t) transformées en droites parallèles de vecteur directeur $\vec{i} + p\vec{t}$.

— les transformations, dites $T_d(O'\vec{i})$, pour lesquelles les images des plans (P_t) sont des plans parallèles au plan P_{i_d}. Cela correspond dans $\mathbf{R}^2$ aux droites (D_t) transformées en droites parallèles de vecteur directeur $\vec{i} - p\vec{t}$.

On définit de même $T_c(O'\vec{j})$ et $T_d(O'\vec{j})$ en inversant les rôles des axes $(O'\vec{i})$ et $(O'\vec{j})$.

Nous devons définir ces différentes transformations. Or, comme nous l'avons constaté au chapitre 3, paragraphe III, l'expression d'une transformation T_R dépend de la représentation R qui la paramètre, c'est-à-dire des valeurs des composantes b_1, b_2, b_3, v de la matrice P et du vecteur V qui définissent R.

Pour décrire les différentes formes possibles d'une transformation effective que nous souhaitons appliquer dans cette méthode, nous devons donc examiner de manière exhaustive les différentes valeurs possibles des données b_1, b_2, b_3 et v associées à R et déterminer, pour chacune de ces valeurs, la forme de la transformation considérée.

Une représentation R est obtenue par application sur la représentation initiale R_0 d'une suite de transformations notée S. Or, l'étude préliminaire du chapitre 3, par. III, a montré que l'expression d'une transformation T_R donnée dépend de la manière dont on a obtenu la représentation R à partir de la repésentation R_0. Plus précisément, elle dépend de la nature de la première transformation de la suite S appliquée à R_0.

Nous donnons donc ci-dessous une définition des tranformations effectives T_R appliquées dans cette méthode en caractérisant une représentation non pas par ses valeurs b_1, b_2, b_3 et v associées, ce qui entrainerait une suite importante de différents cas, mais par la nature de la première transformation appliquée à la représentation initiale R_0 pour obtenir la représentation R.

Pour cela, nous notons $Prem(O'\vec{x})$, la première transformation de la suite S en référence à l'axe $(O'\vec{x})$. Si la suite S ne contient aucune transformation en référence à l'axe $(O'\vec{x})$, nous notons $Prem(O'\vec{x}) = Id$ où Id est l'application identité.

On peut alors définir les transformations effectives de la manière suivante :

DÉFINITION 6. — Soit R une représentation définie par

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix} \quad \text{avec} \quad b_3 \neq 0$$

Les transformations, appliquées à la représentation R, qui sont utilisées dans la méthode, sont de la forme

$$\begin{array}{ccc} \mathbf{R}^3 & \rightarrow & \mathbf{R}^5 \\ (i,j,t)_{E.T.} & \rightarrow & (i,j,t')_{E.T.} \end{array}$$

où la valeur de t' est :

pour $T_c(O'\vec{i})$

si $Prem(O'\vec{i}) \in \{Id, T_c(O'\vec{i})\}$ $\quad t' = i + t$

si $Prem(O'\vec{i}) = T_d(O'\vec{i})$ $\quad t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v$

pour $T_c(O'\vec{j})$

si $Prem(O'\vec{j}) \in \{Id, T_c(O'\vec{j})\}$ $\quad t' = j + t$

si $Prem(O'\vec{j}) = T_d(O'\vec{j})$ $\quad t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v$

pour $T_d(O'\vec{i})$

si $Prem(O'\vec{i}) \in \{Id, T_d(O'\vec{i})\}$ $\quad t' = -i + t + i_{max} - i_{min}$

si $Prem(O'\vec{i}) = T_c(O'\vec{i})$ $\quad t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v$

pour $T_d(O'\vec{j})$

si $Prem(O'\vec{j}) \in \{Id, T_d(O'\vec{j})\}$ $\quad t' = -j + t + j_{max} - j_{min}$

si $Prem(O'\vec{j}) = T_c(O'\vec{j})$ $\quad t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v$

où i_{min} et i_{max} sont les bornes respectivement inférieure et supérieure de l'indice i sur le domaine D, j_{min} et j_{max} celles de l'indice j.

REMARQUE 1. — Par définition de D, la valeur de i_{max} peut être infinie. Lorsque c'est le cas, la transformation $T_d(O'\vec{i})$ avec $Prem(O'\vec{i}) = Id$, ne peut donc pas être appliquée.

REMARQUE 2. — Les transformations décrites ci-dessus déplacent les points dans l'espace-temps parallèlement à l'axe des temps $(O'\vec{t})$, comme nous l'avons exposé dans la présentation générale de la méthode au chapitre 3.

REMARQUE 3. — Dans $\mathbf{R}^2$, les transformations se font toujours en référence à l'axe $(O'\vec{i})$.

REMARQUE 4. — Notons que la forme des quatre transformations dans les deuxièmes cas est la même :

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,t - \frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{1}{b_3}t - \frac{1}{b_3}v) \end{array}$$

Or :

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix}_{B.C.} = P^{-1}\begin{pmatrix} i \\ j \\ t \end{pmatrix}_{E.T.} - P^{-1}V = \begin{pmatrix} i \\ j \\ -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{1}{b_3}t - \frac{1}{b_3}v \end{pmatrix}_{B.C.}$$

Une autre définition de cette transformation est donc :

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,t+k)_{E.T.} \end{array}$$

où k est la composante du point de coordonnées $(i,j,t)_{E.T.}$ sur le vecteur $\vec{b_k}$ de la base canonique, c'est-à-dire l'indice de récurrence du calcul élémentaire associé au point considéré.

Cela signifie que l'effet de cette transformation est d'"écarter" les hyperplans constitués des points de même indice de récurrence dans la base canonique.
Dans la suite, nous appelons ce type de transformation T_{ec}.

En appliquant le théorème 1 aux différentes transformations de la définition 6, on obtient immédiatement les résultats suivants.

PROPRIÉTÉ 3. — *Soit R une représentation définie par la matrice de passage P et le vecteur de translation V avec*

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

Son image par une transformation est une représentation $T_R(R)$ définie par :

— *pour $T_c(O'\vec{i})$ quand $Prem(O'\vec{i}) \in \{Id, T_c(O'\vec{i})\}$*

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1+1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

— *pour $T_c(O'\vec{j})$ quand $Prem(O'\vec{j}) \in \{Id, T_c(O'\vec{j})\}$*

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2+1 & b_3 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

— *pour $T_d(O'\vec{i})$ quand $Prem(O'\vec{i}) \in \{Id, T_d(O'\vec{i})\}$*

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1-1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ v + i_{max} - i_{min} \end{pmatrix}$$

— *pour $T_d(O'\vec{j})$ quand $Prem(O'\vec{j}) \in \{Id, T_d(O'\vec{j})\}$*

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2-1 & b_3 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ v + j_{max} - j_{min} \end{pmatrix}$$

— *pour T_{ec}, c'est-à-dire $T_y(O'\vec{x})$ quand $Prem(O'\vec{x}) = T_{\overline{y}}(O'\vec{x})$ avec $y \in \{c,d\}$ et $x \in \{i,j\}$*

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3+1 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

La représentation initiale est définie par $b_1 = b_2 = 0$, $b_3 = 1$ et $v = 0$. Nous pouvons donc, en tenant compte de cette remarque et de la propriété 3, définir les préconditions sur $Prem(O'\vec{i})$ ou $Prem(O'\vec{j})$ par des conditions équivalentes sur les valeurs de b_1, b_2, b_3 et v caractérisant la représentation R considérée. On a les résultats suivants :

$$\begin{array}{lcll} Prem(O'\vec{i}) & = & Id & \Leftrightarrow \; b_1 = 0 \\ Prem(O'\vec{i}) & = & T_c(O'\vec{i}) & \Leftrightarrow \; b_1 > 0 \\ Prem(O'\vec{i}) & = & T_d(O'\vec{i}) & \Leftrightarrow \; b_1 < 0 \\ Prem(O'\vec{j}) & = & Id & \Leftrightarrow \; b_2 = 0 \\ Prem(O'\vec{j}) & = & T_c(O'\vec{j}) & \Leftrightarrow \; b_2 > 0 \\ Prem(O'\vec{j}) & = & T_d(O'\vec{j}) & \Leftrightarrow \; b_2 < 0 \end{array}$$

Nous pouvons donc résumer l'effet des différentes transformations sur les valeurs b'_1, b'_2, b'_3, v' associées à la représentation image par le tableau 4.1.

REMARQUE. — Nous constatons que les valeurs de b_1, b_2, b_3 associées à une représentation R indiquent la nature des transformations appliquées sur la représentation initiale pour obtenir la représentation R :

$b_1 = 0$: aucune transformation en référence à $(O'\vec{i})$, $Prem(O'\vec{i}) = Id$

	Condition avant transformation	Effet de la transformation
$T_c(O'\vec{i})$	$b_1 \geq 0$	$b'_1 \leftarrow b_1 + 1$
	$b_1 < 0$	$b'_3 \leftarrow b_3 + 1$
$T_c(O'\vec{j})$	$b_2 \geq 0$	$b'_2 \leftarrow b_2 + 1$
	$b_2 < 0$	$b'_3 \leftarrow b_3 + 1$
$T_d(O'\vec{i})$	$b_1 \leq 0$	$b'_1 \leftarrow b_1 - 1$ $v' \leftarrow v + i_{max} - i_{min}$
	$b_1 > 0$	$b'_3 \leftarrow b_3 + 1$
$T_d(O'\vec{j})$	$b_2 \leq 0$	$b'_2 \leftarrow b_2 - 1$ $v' \leftarrow v + j_{max} - j_{min}$
	$b_2 > 0$	$b'_3 \leftarrow b_3 + 1$

TABLEAU 4.1. Effet de l'application d'une transformation

$b_1 > 0$: b_1 transformations $T_c(O'\vec{i})$ avec $Prem(O'\vec{i}) = T_c(O'\vec{i})$

$b_1 < 0$: $-b_1$ transformations $T_d(O'\vec{i})$ avec $Prem(O'\vec{i}) = T_d(O'\vec{i})$

$b_2 = 0$: aucune transformation en référence à $(O'\vec{j})$, $Prem(O'\vec{j}) = Id$

$b_2 > 0$: b_2 transformations $T_c(O'\vec{j})$ avec $Prem(O'\vec{j}) = T_c(O'\vec{j})$

$b_2 < 0$: $-b_2$ transformations $T_d(O'\vec{j})$ avec $Prem(O'\vec{j}) = T_d(O'\vec{j})$

$b_3 = 1$: aucune transformation de type T_{ec}

$b_3 > 1$: $b_3 - 1$ transformations de type T_{ec}, sans distinction de celles effectuées en référence à $(O'\vec{i})$ et $(O'\vec{j})$

Nous pouvons alors donner une définition des transformations effectives appliquées portant directement sur les valeurs de b_1, b_2, b_3, v qui caractérisent la représentation R, paramétrant une transformation. Cette définition est équivalente à la définition 6.

DÉFINITION 7. — Soit une représentation R définie par

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad et \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix} \quad b_3 \neq 0$$

Les transformations, appliquées à la représentation R, qui sont utilisées dans la méthode, sont de la forme

$$\begin{array}{ccc} \mathbf{R^3} & \rightarrow & \mathbf{R^3} \\ (i,j,t)_{E.T.} & \rightarrow & (i,j,t')_{E.T.} \end{array}$$

où la valeur de t' est :

$$\begin{array}{llll}
\text{pour } T_c(O'\vec{i}) & \text{si } b_1 \geq 0 & t' = i + t \\
& \text{si } b_1 < 0 & t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v \\
\text{pour } T_c(O'\vec{j}) & \text{si } b_2 \geq 0 & t' = j + t \\
& \text{si } b_2 < 0 & t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v \\
\text{pour } T_d(O'\vec{i}) & \text{si } b_1 \leq 0 & t' = -i + t + i_{max} - i_{min} \\
& \text{si } b_1 > 0 & t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v \\
\text{pour } T_d(O'\vec{j}) & \text{si } b_2 \leq 0 & t' = -j + t + j_{max} - j_{min} \\
& \text{si } b_2 > 0 & t' = -\frac{b_1}{b_3}i - \frac{b_2}{b_3}j + \frac{b_3+1}{b_3}t - \frac{1}{b_3}v
\end{array}$$

La propriété suivante indique comment sont modifiées les composantes d'un vecteur générateur sur l'espace-temps par application d'une transformation.

PROPRIÉTÉ 4. — *Soit une représentation R d'un problème définie par*

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad et \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

Pour cette représentation, un vecteur générateur $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ a pour coordonnées dans l'espace-temps

$$\Phi_d = (\varphi_i, \varphi_j, \varphi_t)_{E.T.} = (\varphi_i, \varphi_j, b_1\varphi_i + b_2\varphi_j + b_3\varphi_k)_{E.T.}$$

Pour la représentation $T_R(R)$, image de la représentation R par une transformation T_R décrite à la définition 5, les coordonnées d'un vecteur générateur dans l'espace-temps sont :

$$\begin{aligned} \Phi_d &= (\varphi_i, \varphi_j, (\alpha + \gamma b_1)\varphi_i + (\beta + \gamma b_2)\varphi_j + \gamma b_3\varphi_k)_{E.T.} \\ &= (\varphi_i, \varphi_j, \alpha\varphi_i + \beta\varphi_j + \gamma\varphi_t)_{E.T.} \end{aligned}$$

Démonstration. — D'après le théorème 1, la représentation $T_R(R)$ est définie par

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha + \gamma b_1 & \beta + \gamma b_2 & \gamma b_3 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ \gamma v + \delta \end{pmatrix}$$

Pour cette représentation, on a donc $\Phi_{d_{E.T.}} = P'\Phi_{d_{B.C.}}$. Notons les composantes de Φ_d pour cette représentation par $(\varphi_i, \varphi_j, \varphi'_t)_{E.T.}$. On a donc :

$$\begin{aligned} \varphi'_t &= (\alpha + \gamma b_1)\varphi_i + (\beta + \gamma b_2)\varphi_j + \gamma b_3\varphi_k \\ &= \alpha\varphi_i + \beta\varphi_j + \gamma(b_1\varphi_i + b_2\varphi_j + b_3\varphi_k) \\ &= \alpha\varphi_i + \beta\varphi_j + \gamma\varphi_t \end{aligned}$$

D'où le résultat énoncé ci-dessus.

Pour les transformations décrites à la définition 6, on obtient, en appliquant la propriété 4, les résultats suivants.

PROPRIÉTÉ 5. — *Soit une représentation R quelconque, définie par*

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad et \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

Soit $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ un vecteur générateur. Notons $(\varphi_i, \varphi_j, \varphi_t)_{E.T.}$ ses coordonnées dans l'espace-temps pour la représentation R. Ses coordonnées dans l'espace-temps, pour la représentation image de R par une transformation sont $(\varphi_i, \varphi_j, \varphi'_t)_{E.T.}$ où φ'_t est défini par :

pour $T_c(O'\vec{i})$ si $Prem(O'\vec{i}) \in \{Id, T_c(O'\vec{i})\}$, $\varphi'_t = \varphi_t + \varphi_i$

pour $T_c(O'\vec{j})$ si $Prem(O'\vec{j}) \in \{Id, T_c(O'\vec{j})\}$, $\varphi'_t = \varphi_t + \varphi_j$

pour $T_d(O'\vec{i})$ si $Prem(O'\vec{i}) \in \{Id, T_d(O'\vec{i})\}$, $\varphi'_t = \varphi_t - \varphi_i$

pour $T_d(O'\vec{j})$ si $Prem(O'\vec{j}) \in \{Id, T_d(O'\vec{j})\}$, $\varphi'_t = \varphi_t - \varphi_j$

pour T_{ec}, $\varphi'_t = \varphi_t + \varphi_k$

Comme nous l'avons exposé dans la présentation intuitive de la méthode au chapitre 3, par. III, nous choisissons de n'appliquer une transformation sur une représentation que s'il existe, sur cette représentation, une condition portant sur l'utilisation d'une donnée.

En effet, du point de vue réalisation d'un réseau systolique par un circuit VLSI, les réseaux les plus intéressants sont les sytoliques purs dans lesquels il n'y a pas de communications globales de données (comme des diffusions), c'est à dire où toutes les données utilisées plusieurs fois dans le réseau le sont à des instants successifs.

C'est pourquoi, dans cette méthode, une transformation T n'est appliquée sur une représentation R que s'il existe, sur R, une condition d'utilisations simultanées d'une donnée d en différents points du domaine. La transformation T est telle que, sur $T_R(R)$, la donnée d est utilisée successivement. Ceci se traduit par :

Condition méthodologique d'application d'une transformation

Une transformation T_R définie par

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\alpha i + \beta j + \gamma t + \delta)_{E.T.} \end{array}$$

est appliquée sur une représentation R si et seulement si il existe un vecteur générateur non nul $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ dont les coordonnées dans l'espace-temps pour la représentation R sont $\Phi_d = (\varphi_i, \varphi_j, \varphi_t)_{E.T.}$ tel que :

$$\varphi_t = 0 \quad et \quad \alpha\varphi_i + \beta\varphi_j + \gamma\varphi_t \neq 0$$

La condition $\varphi_t = 0$ traduit le fait que la donnée d est utilisée simultanément sur une représentation R. La condition $\alpha\varphi_i + \beta\varphi_j + \gamma\varphi_t \neq 0$ traduit le fait que la donnée d n'est plus utilisée simultanément sur la représentation $T_R(R)$ puisque $\alpha\varphi_i + \beta\varphi_j + \gamma\varphi_t$ est la composante du vecteur générateur sur l'axe des temps pour la représentation $T_R(R)$ (*cf.* propriété 4).

IV. Relation de précédence de représentations

DÉFINITION 8. — Soit $C_R(t_0)$ l'ensemble des points de la représentation correspondant aux calculs exécutables simultanément à l'instant t_0. Cet ensemble est défini par

$$C_R(t_0) = \{(i,j,k)_{B.C.} \in D \quad | \quad b_1 i + b_2 j + b_3 k + v = t_0\}$$

où b_1, b_2, b_3, v sont les valeurs liées à la représentation R.

Cette définition exprime le fait qu'un point $(i,j,k)_{B.C.}$ correspond à un point de l'espace-temps défini par :

$$\begin{pmatrix} i \\ j \\ t \end{pmatrix}_{E.T.} = P \begin{pmatrix} i \\ j \\ k \end{pmatrix}_{B.C.} + V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

$$= \begin{pmatrix} i \\ j \\ b_1 i + b_2 j + b_3 k + v \end{pmatrix}$$

d'où $t = b_1 i + b_2 j + b_3 k + v$.

DÉFINITION 9. — On dit que la représentation R *précède* la représentation R', ce que l'on note $R < R'$, si et seulement si

$$\exists \lambda, \mu \in \mathbf{N}, \lambda > 1 \quad \textit{tels que} \quad \forall t \geq 0, \quad C_R(t) = C_{R'}(\lambda t + \mu)$$

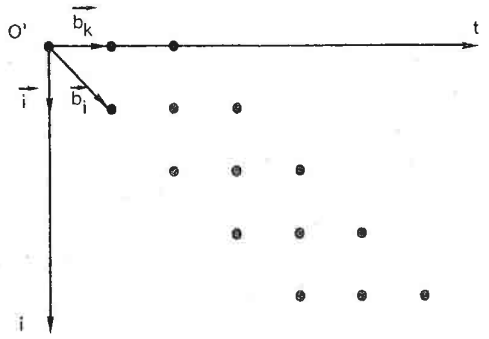


Figure 4.2 $R = T_c(O\vec{i})_{R_0}(R_0)$

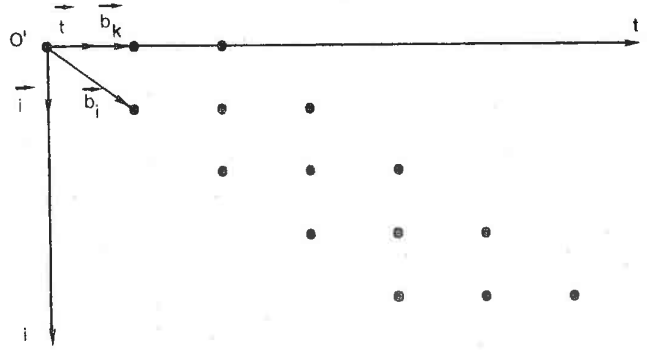


Figure 4.3 $R' = (T_c(O'\vec{i}) \circ T_d(O'\vec{i}) \circ T_c(O'\vec{i}))_{R_0}(R_0)$

EXEMPLE. — Considérons un problème défini sur $\mathbf{R}^2$ avec pour domaine

$$D = \{(i,k) \in \mathbf{Z}^2 \quad | \quad 0 \leq i \leq 4 \quad 0 \leq k \leq 2\}$$

Faisons abstraction des données du problème et considérons les deux représentations $R = T_c(O\vec{i})_{R_0}(R_0)$ et $R' = (T_c(O'\vec{i}) \circ T_d(O'\vec{i}) \circ T_c(O'\vec{i}))_{R_0}(R_0)$ visualisées figures 4.2 et 4.3 respectivement.

Les calculs simultanés des deux représentations R et R' sont identiques : $C_R(t) = C_{R'}(2t)$. La représentation R précède donc la représentation R'.

PROPRIÉTÉ 6. — *La relation de précédence ainsi définie est une relation d'ordre strict partiel.*

Démonstration. — C'est une relation d'ordre partiel car toutes les représentations ne sont pas en relation. Ainsi, la représentation initiale R_0 n'est en relation avec aucune autre représentation.
C'est une relation d'ordre strict. Elle est :
— non réflexive car $\lambda > 1$
— antisymétrique. En effet, soient deux représentations quelconques R et R' telles que $R < R'$ et$R' < R$, c'est-à-dire

$$\exists \lambda, \mu, \lambda', \mu' \in \mathbf{N} \quad \textit{tels que} \quad \forall t \geq 0, \quad C_R(t) = C_{R'}(\lambda t + \mu)$$

$$\textit{et} \quad C_{R'}(t) = C_R(\lambda' t + \mu')$$

On a alors $\forall t \geq 0$,

$$C_R(t) = C_{R'}(\lambda t + \mu) = C_R(\lambda'(\lambda t + \mu) + \mu') = C_R(\lambda\lambda' t + \lambda' m + \mu')$$

Donc $\forall t \geq 0, \quad C_R(t) = C_R(\lambda\lambda' t + \lambda' m + \mu')$.
Cette égalité est vraie si et seulement si

$$\lambda\lambda' = 1, \lambda'\mu + \mu' = 0 \Leftrightarrow \lambda = \lambda' = 1, \mu = \mu' = 0$$

On a donc $\forall t \geq 0, \quad C_R(t) = C_{R'}(t) \Leftrightarrow R = R'$

— transitive. En effet, soient trois représentations R, R' et R'' telles que $R < R'$ et $R' < R''$, c'est-à-dire

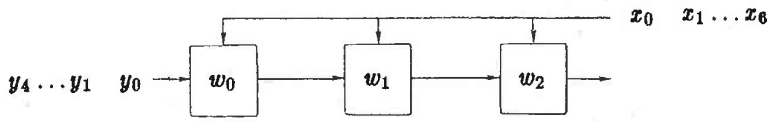
$$\exists \lambda', \lambda'', \mu', \mu'' \in \mathbf{N}, \lambda' > 1, \lambda'' > 1 \quad \text{tels que}$$

$$C_R(t) = C_{R'}(\lambda' t + \mu') \quad \text{et} \quad C_{R'}(t) = C_{R''}(\lambda'' t + \mu'')$$

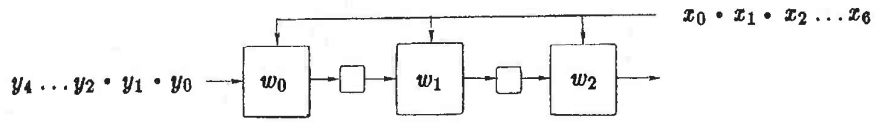
On a alors $\forall t \geq 0$,

$$\begin{aligned} C_R(t) &= C_{R'}(\lambda' t + \mu') \\ &= C_{R''}(\lambda''(\lambda' t + \mu') + \mu'') \\ &= C_{R''}(\lambda'\lambda'' t + \lambda''\mu' + \mu'') \end{aligned}$$

Posons $\lambda = \lambda'\lambda'', \mu = \lambda''\mu' + \mu''$.
On a $\forall t \geq 0, \quad C_R(t) = C_{R''}(\lambda t + \mu)$, donc $R < R''$.



(a) pour la représentation $R = T_c(O'\vec{i})_{R_0}(R_0)$



(b) pour la représentation $R' = (T_c(O'\vec{i}) \circ T_d(O'\vec{i}) \circ T_c(O'\vec{i}))_{R_0}(R_0)$

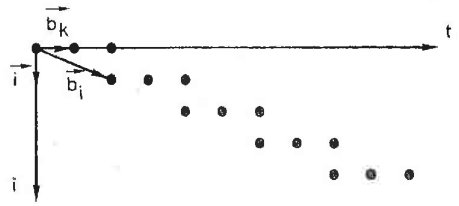
Figure 4.4. Réseaux obtenus pour $\vec{\xi} = (0,1)_{B.C.}$

Cette définition, R précède R', signifie que les deux représentations R et R' ont les mêmes calculs simultanés et que deux calculs successifs sont séparés par plus de cycles de temps dans R' que dans R. De telles représentations conduisent donc à des réseaux systoliques similaires. Ceux issus de R' ont un temps total d'exécution supérieur à celui correspondant aux réseaux issus de R en raison de la présence de cellules de retard sur les flots. Ces cellules traduisent les cycles séparant deux calculs successifs.

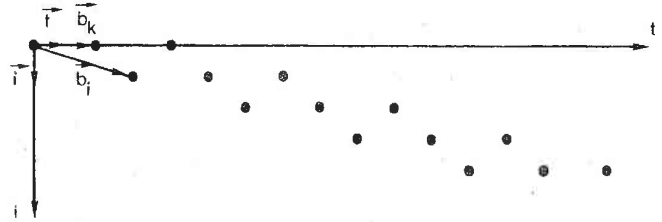
L'intérêt de cette relation est donc le suivant : supposons que la représentation R' nouvellement obtenue soit telle qu'il existe une représentation R préalablement obtenue telle que $R < R'$. Il est alors inutile de chercher les réseaux systoliques correspondants à la représentation R' puisqu'ils sont similaires à ceux déjà obtenus à partir de la représentation R.

EXEMPLE. — Considérons à nouveau les deux représentations décrites à l'exemple précédent, associées au problème du produit de convolution.

Si l'on choisit une même direction d'allocation, $\vec{\xi} = (1,0)_{B.C.}$, on obtient les réseaux de la figure 4.4.



$T_c(O'\vec{i})_R(R)$



$T_c(O'\vec{i})_{R'}(R')$

FIGURE 4.5. Représentations images de R et R' par $T_c(O'\vec{i})$

Le réseau associé à R' correspond à celui associé à R où ont été ajouté des cellules de retard et des espacements des données sur les flots. Il ont les mêmes caractéristiques physiques (nombre de cellules de calcul, nature des différents flots). Le temps total d'exécution du deuxième réseau est le double de celui du premier. Le réseau associé à R' ne présente donc pas d'intérêt.

REMARQUE. — Soient deux représentations R et R' telles que $R < R'$. Si l'on applique une même transformation T sur ces deux représentations, les représentations obtenues $T_R(R)$ et $T_{R'}(R')$ ne sont , en général, pas en relation de précédence : $R < R' \not\Leftrightarrow T_R(R) < T_{R'}(R')$.

Ainsi, pour l'exemple ci-dessus : supposons que l'on puisse appliquer la même transformation $T_c(O'\vec{i})$ sur les deux représentations R et R'. Cela suppose que la condition méthodologique d'application d'une transformation est vérifiée sur les deux représentations. Les représentations images obtenues sont visualisées figure 4.5. Elles ne sont manisfestement pas en relation.

THÉORÈME 2. — *Soit S une suite de transformations appliquée sur la représentation initiale R_0. La représentation image obtenue, $S_{(R_0)}(R_0)$, est notée R. Soit T une suite de transformations appppliquée sur la représentation R. La représentation image de R par T_R est $T_R(R) = (T \circ S)_{R_0}(R_0)$. Elle est notée R_t.*

On a $R < R_t$ si et seulement si T est une R-affinité définie par

$$\begin{array}{rccc} T: & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\lambda t+\mu)_{E.T.} \end{array}$$

avec $\lambda, \mu \in \mathbf{N}, \lambda > 1$.

Comme précédemment, nous employons le terme de R-affinité pour indiquer que T est une affinité paramétrée par la représentation R. En effet, les valeurs des constantes λ et μ dépendent des composantes de P et V liées à R.

Démonstration. — Supposons la représentation R définie par

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

et la suite de transformations T_{R_S} définie par

$$\begin{array}{rccc} T_R: & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\alpha i+\beta j+\gamma t+\delta)_{E.T.} \end{array}$$

La représentation R_t image de la représentation R par T_R est donc définie, d'après le théorème 1, par

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha+\beta b_1 & \beta+\gamma b_2 & \gamma b_3 \end{pmatrix} \quad \text{et} \quad V' = \begin{pmatrix} 0 \\ 0 \\ \gamma v+\delta \end{pmatrix}$$

D'après la définition 9, on a :

$$R < R_t \Leftrightarrow$$

$$\exists \lambda, \mu \in \mathbf{N}, \lambda > 1 \quad \text{tels que} \quad \forall t \geq 0, \quad C_R(t) = C_{R_t}(\lambda t + \mu) \qquad (1)$$

Or, d'après la définition 8, on a :

$$C_R(t) = \{(i,j,k)_{B.C.} \in D \quad | \quad b_1 i + b_2 j + b_3 k + v = t\}$$

$$C_{R_t}(\lambda t + \mu) = \{(i,j,k)_{B.C.} \in D \quad | \quad (\alpha + \gamma b_1)i + (\beta + \gamma b_2)j + \gamma b_3 k + \gamma v + \delta = \lambda t + \mu\}$$

d'où

$$(1) \Leftrightarrow \forall (i,j,k) \in D,$$

$$\lambda(b_1 i + b_2 j + b_3 k + v) + \mu = (\alpha + \gamma b_1)i + (\beta + \gamma b_2)j + \gamma b_3 k + \gamma v + \delta$$

$$\Leftrightarrow \begin{cases} \lambda b_1 = \alpha + \gamma b_1 \\ \lambda b_2 = \beta + \gamma b_2 \\ \lambda b_3 = \gamma b_3 \\ \lambda v + \mu = \gamma v + \delta \end{cases}$$

$$\Rightarrow \begin{cases} \lambda = \gamma \\ \alpha = 0 \\ \beta = 0 \\ \mu = \delta \end{cases}$$

La transformation T_R est donc définie par

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \to & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \to & (i,j,\lambda t + \mu)_{E.T.} \end{array}$$

EXEMPLE. — Reprenons l'exemple précédent où l'on considère les deux représentations
$R = T_c(O'\vec{i})_{R_0}(R_0)$ et
$R' = (T_c(O'\vec{i}) \circ T_d(O'\vec{i}) \circ T_c(O'\vec{i}))_{R_0}(R_0) = (T_c(O'\vec{i}) \circ T_d(O'\vec{i}))_R(R)$.
La représentation R est définie par $P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ et $V = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. La représentation R' est obtenue, à partir de la représentation $R = T_c(O'\vec{i})_{R_0}(R_0)$, en appliquant la suite de transformations $T = T_c(O'\vec{i}) \circ T_d(O'\vec{i})$ définies par

$$\begin{array}{rccc} T_d(O'\vec{i}) : & \mathbf{R}^2 & \to & \mathbf{R}^2 \\ & (i,t)_{E.T.} & \to & (i,t+k)_{E.T.} \end{array}$$

car $Prem(O'\vec{i}) = T_c(O'\vec{i})$(*cf.* remarque 4 de la définition 6).

$$\begin{array}{rccc} T_c(O'\vec{i}) : & \mathbf{R}^2 & \to & \mathbf{R}^2 \\ & (i,t)_{E.T.} & \to & (i,i+t)_{E.T.} \end{array}$$

où k est défini par $\begin{pmatrix} i \\ k \end{pmatrix}_{B.C.} = P^{-1} \times \begin{pmatrix} i \\ t \end{pmatrix}_{E.T.} - P^{-1} \times V$,
c'est-à-dire $k = -i + t$. La transformation $T = T_c(O'\vec{i}) \circ T_d(O'\vec{i})$ est donc définie par

$$\begin{array}{rccccc} T : & \mathbf{R}^2 & \to & \mathbf{R}^2 & \to & \mathbf{R}^2 \\ & (i,t)_{E.T.} & \to & (i,t+k) = (i,2t-i) & \to & (i,i+(2t-i)) = (i,2t) \end{array}$$

La transformation T est bien une R-affinité, on a donc $R < R'$.

Supposons la représentation R caractérisée par les valeurs b_1, b_2, b_3 et v, c'est-à-dire définie par

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad et \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

Considérons une transformation T_R définie par

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \to & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \to & (i,j,\alpha i + \beta j + \gamma t + \delta)_{E.T.} \end{array}$$

D'après le théorème 1, la représentation R a pour image par la transformation T_R, la représentation $R_t = T_R(R)$ définie par

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b'_1 & b'_2 & b'_3 \end{pmatrix} \quad et \quad V' = \begin{pmatrix} 0 \\ 0 \\ v' \end{pmatrix}$$

avec $b'_1 = \alpha + \gamma b_1, b'_2 = \beta + \gamma b_2, b'_3 = \gamma b_3, v' = \gamma v + \delta$.

La transformation T est une R-affinité si et seulement si $\alpha = \beta = 0$. Cette condition s'exprime directement au niveau des représentations par

$$\begin{cases} b'_1 = \gamma b_1 \\ b'_2 = \gamma b_2 \end{cases} \Leftrightarrow \begin{cases} b'_1 b_3 = b'_3 b_1 \\ b'_2 b_3 = b'_3 b_2 \end{cases}$$

car $\gamma = b'_3/b_3$.

REMARQUE 1. — Cette condition ne peut être satisfaite pour la représentation initiale où $b_1 = b_2 = 0$ et $b_3 = 1$. Cela signifie que la représentation initiale n'est en relation de précédence avec aucune autre représentation.

REMARQUE 2. — Si $b'_3 = b_3$ alors $b'_1 = b_1$ et $b'_2 = b_2$. La transformation T doit donc être telle que $b'_3 \neq b_3$. Pour cela, il faut qu'au moins une des transformations $T_l, l = 1, \ldots, q$, composant T soit de type T_{ec} (*cf.* la propriété 3 et le tableau déduit de cette propriété).

REMARQUE 3. — Si $b_1 = 0$ (resp. $b_2 = 0$) alors $b'_1 = 0$ (resp. $b'_2 = 0$) car $b_3 \neq 0$. Cette constatation signifie que si la suite S ne contient aucune transformation sur $(O'\vec{i})$, c'est-à-dire si $b_1 = 0$ (resp. $(O'\vec{j})$), alors la suite T n'en contient pas non plus. T est donc composée d'une suite de transformations sur $(O'\vec{j})$ (resp. $(O'\vec{i})$) vérifiant $b'_2 b_3 = b'_3 b_2$.

PROPRIÉTÉ 7. — *Soit $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ un vecteur générateur de coordonnées $(\varphi_i, \varphi_j, \varphi_t)_{E.T.}$ dans l'espace-temps pour une représentation R donnée. Soit T_R une R-affinité définie par*

$$\begin{array}{rccc} T_R : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\lambda t + \mu)_{E.T.} \end{array}$$

avec $\lambda, \mu \in \mathbf{N}, \lambda > 1$.

Les coordonnées de ce vecteur générateur dans l'espace-temps pour la représentation $T_R(R)$, image de R par T_R, sont

$$\Phi_d = (\varphi_i, \varphi_j, \lambda\varphi_t)_{E.T.}$$

Démonstration. — Elle se déduit immédiatement de la propriété 4.

Dans la méthode proposée, une transformation ne sera appliquée sur une représentation que si la condition méthodologique sur l'utilisation des données du problème est vérifiée. Cette condition exprime que

— lorsqu'une utilisation simultanée des occurrences d'une donnée est vérifiée sur une représentation R quelconque, on peut appliquer sur R une transformation.Elle conduit à une autre représentation, soit nouvelle, soit en relation de précédence avec une représentation obtenue précédemment.

— aucune transformation ne sera appliquée dès qu'il n'y a plus de contraintes de simultanéité de certaines occurrences de données.

Pour garantir la terminaison du processus de construction des réseaux systoliques, il faut vérifier que le nombre de transformations applicables à partir de la représentation initiale est fini. D'où le théorème suivant.

THÉORÈME 3. — *Sous l'hypothèse qu'une transformation ne peut être appliquée sur une représentation R quelconque que si une contrainte de simultanéité existe sur R, l'ensemble des transformations applicables, pour tout problème systolisable, à partir de la représentation initiale, est fini. L'ensemble des représentations est donc fini.*

Nous utilisons les lemmes suivants pour la démonstration de ce théorème.

LEMME 1. — *Soient R_1 et R_2 deux représentations. Soit S une suite de transformations appliquée à R_1 telle que $R_2 = S_{R_1}(R_1)$. Si les représentations R_1 et R_2 présentent une simultanéité due au même vecteur générateur, c'est-à-dire s'il existe un vecteur $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ tel que $\varphi_{t(R_1)} = \varphi_{t_{(R_2)}} = 0$, alors $R_1 < R_2$.*

Démonstration. — D'après le théorème 2, on a $R_1 < R_2$ si et seulement si S est une R_1-affinité définie par :

$$\begin{array}{rccc} S : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\lambda t + \mu)_{E.T.} \end{array}$$

avec $\lambda, \mu \in \mathbf{N}, \lambda > 1$.

Il faut donc montrer que S est une R_1-affinité.
Supposons la représentation R_1 définie par :

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

et la suite de transformations S appliquée à R_1 définie par :

$$\begin{array}{rccc} S : & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\alpha i + \beta j + \gamma t + \delta)_{E.T.} \end{array}$$

Notons $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$ le vecteur générateur qui crée la simultanéité sur R_1 et R_2. D'après la propriété 4, on a donc :

$$\begin{cases} \varphi_{t_{(R_1)}} = b_1\varphi_i + b_2\varphi_j + b_3\varphi_k = 0 \\ \varphi_{t_{(R_2)}} = \alpha\varphi_i + \beta\varphi_j + \gamma\varphi_{t_{(R_1)}} = 0 \end{cases}$$

$\Rightarrow \alpha\varphi_i + \beta\varphi_j = 0$
Or les valeurs de φ_i et φ_j peuvent être quelconques. Donc $\alpha = \beta = 0$. La suite de transformations S est bien une R_1-affinité donc $R_1 < R_2$.

LEMME 2. — *Soient deux représentations R_1 et R_2 telles que $R_1 < R_2$. Soit T une transformation telle que, pour la représentation $T(R_1)$, il y a une simultanéité d'utilisation d'une donnée, c'est-à-dire qu'il existe un vecteur générateur $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.G.}$ dont la composante temporelle pour $T(R_1)$, $\varphi_{t_{T(R_1)}}$ soit nulle.*

La représentation $T(R_2)$ est telle qu'il n'y a pas de simultanéité d'utilisations de données, c'est-à-dire que tout vecteur générateur Φ_d est tel que sa composante temporelle pour $T(R_2)$, $\varphi_{t_{T(R_2)}}$ est non nulle.

Démonstration. — D'après le théorème 2, comme $R_1 < R_2$, il existe une suite de transformations S définie par :

$$\begin{array}{rccc} S: & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\lambda t+\mu)_{E.T.} \end{array}$$

avec $\lambda, \mu \in \mathbf{N}, \lambda > 1$ telle que $R_2 = S(R_1)$.

Considérons un vecteur générateur quelconque $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.G.}$ et notons $\varphi_{t_{(R_1)}}$ sa composante temporelle pour la représentation R_1.

D'après la propriété 7, sa composante temporelle pour la représentation $R_2 = S(R_1)$ est $\varphi_{t_{(R_2)}} = \lambda \varphi_{t_{(R_1)}}$.

Supposons la transformation T définie par :

$$\begin{array}{rccc} T: & \mathbf{R}^3 & \rightarrow & \mathbf{R}^3 \\ & (i,j,t)_{E.T.} & \rightarrow & (i,j,\alpha i+\beta j+\gamma t+\delta)_{E.T.} \end{array}$$

La transformation T appliquée aux représentations R_1 et R_2 a la même forme dans les deux cas car la précondition portant sur la nature de la première transformation appliquée ($Prem$) est la même pour R_1 et R_2 (puisque $R_1 < R_2$).

D'après la propriété 4 on a :

$$\begin{cases} \varphi_{t_{(T(R_1))}} = \alpha\varphi_i + \beta\varphi_j + \gamma\varphi_{t_{(R_1)}} \\ \\ \varphi_{t_{(T(R_2))}} = \alpha\varphi_i + \beta\varphi_j + \gamma\varphi_{t_{(R_2)}} = \alpha\varphi_i + \beta\varphi_j + \gamma\lambda\varphi_{t_{(R_1)}} \end{cases}$$

— (1) Considérons le vecteur générateur, noté Φ_d^0, qui crée une simultanéité sur $T(R_1)$. Il n'en créait pas sur la représentation R_1 car la

condition méthodologique d'application d'une transformation est telle qu'elle doit lever la simultanéité. On a donc :

$$\begin{cases} \varphi^0_{t_{(T(R_1))}} = \alpha\varphi^0_i + \beta\varphi^0_j + \gamma\varphi^0_{t_{(R_1)}} = 0 \\ \\ \varphi^0_{t_{(R_1)}} \neq 0 \end{cases}$$

Montrons que ce vecteur ne crée pas de simultanéité sur $T(R_2)$. On a :

$$\begin{aligned} \varphi^0_{t_{(T(R_2))}} &= \alpha\varphi^0_i + \beta\varphi^0_j + \gamma\lambda\varphi^0_{t_{(R_1)}} \\ &= \alpha\varphi^0_i + \beta\varphi^0_j + \gamma\varphi^0_{t_{(R_1)}} + \gamma(\lambda-1)\varphi^0_{t_{(R_1)}} \\ &= \gamma(\lambda-1)\varphi^0_{t_{(R_1)}} \end{aligned}$$

Or $\gamma \neq 0$ pour toutes les transformations effectives de la définition 6, $\lambda > 1$ et $\varphi^0_{t_{(R_1)}} \neq 0$.
Donc $\gamma(\lambda-1)\varphi^0_{t_{(R_1)}} \neq 0 \Rightarrow \varphi^0_{t_{(T(R_2))}} \neq 0$.

Le vecteur générateur Φ^0_d ne définit donc pas de simultanéité d'utilisations de données d sur la représentation $T(R_2)$.

— (2) Considérons les vecteurs générateurs, notés $\Phi^l_d, l \neq 0$, qui ne créent pas de simultanéité sur $T(R_1)$.
On a donc $\varphi^l_{t_{(T(R_1))}} = \alpha\varphi^l_i + \beta\varphi^l_j + \gamma\varphi^l_{t_{(R_1)}} \neq 0$.
Supposons $\varphi^l_{t_{(T(R_2))}} = 0$
On a :

$$\begin{aligned} \varphi^l_{t_{(T(R_2))}} &= \alpha\varphi^l_i + \beta\varphi^l_j + \gamma\lambda\varphi^l_{t_{(R_1)}} \\ &= \lambda(\alpha\varphi^l_i + \beta\varphi^l_j + \gamma\varphi^l_{t_{(R_1)}}) - (\lambda-1)(\alpha\varphi^l_i + \beta\varphi^l_j) \\ &= \lambda\varphi^l_{t_{(T(R_1))}} - (\lambda-1)(\alpha\varphi^l_i + \beta\varphi^l_j) \end{aligned}$$

avec $\lambda > 1$. Donc $\varphi^l_{t_{(T(R_2))}} = 0 \Leftrightarrow \varphi^l_{t_{(T(R_1))}} = \frac{\lambda-1}{\lambda}(\alpha\varphi^l_i + \beta\varphi^l_j)$
Or on a $\varphi^l_{t_{(T(R_1))}} \in \mathbf{Z}$ et $\frac{\lambda-1}{\lambda}(\alpha\varphi^l_i + \beta\varphi^l_j) \notin \mathbf{Z}$

Il y a une contradiction. L'hypothèse $\varphi^l_{t_{(T(R_2))}} = 0$ est donc fausse. Les vecteurs générateurs $\Phi^l_d, l \neq 0$, sont tels que $\varphi^l_{t_{(T(R_2))}} \neq 0$. Ils ne définissent donc pas de simultanéité d'utilisations de données sur $T(R_2)$.

La représentation $T(R_2)$ ne contient donc aucune utilisation simultanée de données.

Démonstration du théorème 3. — Par récurrence sur le nombre de vecteurs générateurs.

Nous ne prenons pas en compte les vecteurs générateurs nuls, associés aux données à utilisation unique, car ils ne peuvent induire de transformations.

Pour un seul vecteur générateur $\Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.}$, le nombre de transformations applicables est fini. En effet, sa composante temporelle, pour la représentation initiale R_0, est $\varphi_{t_{(R_0)}} = \varphi_k$.

D'après la condition méthodologique, une transformation T n'est applicable sur R_0 que si $\varphi_{t_{(R_0)}} = 0$ et on a alors $\varphi_{t_{(T(R_0))}} \neq 0$.

Donc, on peut appliquer au plus une transformation sur R_0 dans le cas d'un seul vecteur générateur.

Considérons l'hypothèse de récurrence suivante : supposons que, pour n vecteurs générateurs, l'ensemble des transformations nécessaires à la suppression de toute simultanéité liées à ces n vecteurs est fini.

Montrons qu'il en est de même pour $n+1$ vecteurs générateurs.

Notons $\Phi^1, \ldots, \Phi^n$, les n premiers vecteurs générateurs et Φ^{n+1} le nouveau vecteur.

Considérons successivement chacune des représentations, notée R_1, sans simultanéité due aux vecteurs $\Phi^l, l = 1 \ldots n$. Notons $\Phi^l = (\varphi_i^l, \varphi_j^l, \varphi_k^l)_{B.C.}$.

La représentation R_1 est définie par

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_1 & b_2 & b_3 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

avec $b_1, b_2, b_3, v \in \mathbb{Z}$.

D'après la propriété 4, on a $\varphi^l_{t_{(R_1)}} = b_1\varphi_i^l + b_2\varphi_j^l + b_3\varphi_k^l$ avec, par hypothèse de récurrence, $\varphi^l_{t_{(R_1)}} \neq 0$ pour $l = 1 \ldots n$.

Considérons le $n+1$-ième vecteur Φ^{n+1} caractérisé par

$$\varphi^{n+1}_{t_{(R_1)}} = b_1\varphi_i^{n+1} + b_2\varphi_j^{n+1} + b_3\varphi_k^{n+1}$$

Deux cas sont à distinguer :

cas 1 : $\varphi^{n+1}_{t_{(R_1)}} \neq 0$

Il n'y a aucune simultanéité sur R_1, donc pas de nouvelles transformations applicables. L'ensemble des transformations pour $n+1$ vecteurs générateurs est dans ce cas fini.

cas 2 : $\varphi^{n+1}_{t_{(R_1)}} = 0$

Il y a simultanéité sur R_1 due au vecteur Φ^{n+1}. Il faut donc appliquer une transformation T pour la supprimer.

Notons R_2 la représentation obtenue : $R_2 = T(R_1)$.

D'après la condition méthodologique d'application d'une transformation, le vecteur Φ^{n+1} ne crée plus de simultanéité sur la représentation R_2, c'est-à-dire $\varphi^{n+1}_{t_{(R_2)}} \neq 0$.

Il faut à nouveau distinguer 2 cas :

— **1** : $\forall l \in \{1 \ldots n\}, \varphi^l_{t_{(R_2)}} \neq 0$

Il n'y a aucune simultanéité sur R_2. L'ensemble des transformations applicables est donc fini.

— **2** : $\exists l \in \{1 \ldots n\}$ tel que $\varphi^l_{t_{(R_2)}} = 0$

La suppression de la simultanéité sur R_1 due à Φ^{n+1} a introduit, sur R_2, une simultanéité due à l'un des n premiers vecteurs.

Or, par hypothèse de récurrence, le nombre de transformations nécessaires pour supprimer les simultanéités dues aux n premiers vecteurs est fini.

Donc, il existe une suite finie non vide de transformations, notée S, telle que la représentation R_3, image de R_2 par S, ne présente plus de simultanéité due aux n premiers vecteurs générateurs. On a alors $\varphi^l_{t_{(R_3)}} \neq 0$ pour $l = 1 \ldots n$.

On distingue deux sous-cas :

(a) Si $\varphi^{n+1}_{t_{(R_3)}} \neq 0$ alors il n'y a plus de simultanéité sur R_3, donc plus de transformations applicables.

(b) Si $\varphi^{n+1}_{t_{(R_3)}} = 0$ alors les représentations R_1 et R_3 présentent une simultanéité due au même vecteur générateur Φ^{n+1}. D'après le lemme 1, on a donc $\bar{R}_1 < \bar{R}_3$.

De plus, la représentation $R_2 = T(R_1)$ présente une simultanéité due à un vecteur $\Phi^l, l = 1 \ldots n$.

D'après le lemme 2, la représentation $T(R_3)$, image de R_3 par cette transformation T, ne présente plus aucune simultanéité due aux n vecteurs.

Donc, une transformation, appliquée à la représentation R_3, supprime toute simultanéité sur la représentation image. Aucune transformation n'est alors applicable. L'ensemble des transformations applicables est donc fini.

V.Détermination des réseaux associés à une représentation

Pour déterminer un réseau associé à une représentation R dans $\mathbf{R}^3$, nous faisons le choix d'une direction d'allocation $\vec{\xi}$, vecteur de coordonnées entières de $\mathbf{R}^3$, défini dans la base canonique du problème. Cette direction définit un ensemble de droites dans l'espace. Tous les points du domaine appartenant à une de ces droites représentent les calculs effectués sur la même cellule.

A partir de cette direction, nous déterminons la fonction d'allocation a de $D \subset \mathbf{R}^3$ dans $\mathbf{R}^2$ qui indique, pour tout point P du domaine D, dans quelle cellule référencée par $a(P)$ le calcul associé à P s'exécute.

Les coordonnées du vecteur $\vec{\xi} = (\xi_1, \xi_2, \xi_3)_{B.C.}$ sont choisies premières entre elles afin qu'à partir d'un point quelconque du domaine D, de coordonnées (i, j, k), appartenant à une droite de direction $\vec{\xi}$, on puisse atteindre, par $\vec{\xi}$, tous les points du domaine appartenant aussi à cette droite, de coordonnées $(i + n\xi_1, j + n\xi_2, k + n\xi_3), n \in \mathbf{Z}$. Dans ce cas, on dit que $\vec{\xi}$ est unimodulaire.

Pour définir la fonction d'allocation, nous déterminons une base de $\mathbf{Z}^3$ contenant le vecteur $\vec{\xi}$. Notons $(\vec{\xi}, \vec{\xi}', \vec{\xi}'')$ cette base. Les points du domaine sont définis par de nouvelles coordonnées entières sur cette base. De plus, tous les points du domaine, qui doivent être exécutées sur la même cellule, ont les mêmes composantes entières sur les directions définies par les vecteurs $\vec{\xi}', \vec{\xi}''$ de cette nouvelle base. Ces composantes peuvent donc définir les coordonnées, dans le plan où le réseau est représenté, de la cellule associée.

La fonction d'allocation définit donc, pour tout point du domaine, ses composantes sur les vecteurs de base $\vec{\xi}', \vec{\xi}''$, c'est-à-dire les coordonnées de la cellule où s'exécute le calcul associé à ce point.

Nous présentons les résultats mathétiques utilisés pour déterminer la fonction d'allocation. On trouve dans [NEW 72] les démonstrations des différents théorèmes énoncés. Ces résultats sont présentés pour n quelconque. Ils sont utilisés dans la suite pour $n = 2$ ou $n = 3$.

VI.1. Résultats mathématiques utilisés. — Considérons l'ensemble des matrices carrées de dimension n à coefficients entiers et d'inverse à coefficients entiers. Ces matrices forment un groupe multiplicatif noté $Gl(n, \mathbf{Z})$. Une matrice de $Gl(n, \mathbf{Z})$ est appelée *matrice unimodulaire.*

THÉORÈME 4. — *Soit $M(n, \mathbf{Z})$ l'ensemble des matrices carrées à coefficients entiers. On a :*

$$M \in Gl(n, \mathbf{Z}) \Leftrightarrow M \in M(n, \mathbf{Z}) \quad et \quad det(M) = \pm 1$$

Considérons les matrices particulières de $Gl(n, \mathbf{Z})$ suivantes :
— (1) $E_{ij}(\lambda) = I + \lambda E_{ij}, i \neq j$ où I est la matrice identité et E_{ij} est la matrice à coefficients tous nuls sauf $e_{ij} = 1$.
— (2)T_{ij} la matrice I où les i-ème et j-ème lignes sont échangées.
— (3)D_i la matrice I où la i-ème ligne est multipliée par -1.

Nous appelons *matrice élémentaire* ou *matrice de manipulation*, et nous notons E, une de ces matrices particulières.

THÉORÈME 5. — *$Gl(n, \mathbf{Z})$ est engendré par les matrices élémentaires, c'est-à-dire que toute matrice unimodulaire est un produit de matrices élémentaires.*

Soit une matrice quelconque $M \in Gl(n, \mathbf{Z})$. Notons L_i (resp. L_j) sa i-ème (resp. j-ème) ligne. Considérons la matrice M' obtenue en multipliant à gauche la matrice M par une matrice élémentaire E et notons L'_i (resp. L'_j) sa i-ème (resp.j-ème) ligne. La matrice $M' = E \times M$ est définie de la manière suivante :
— si $E = E_{ij}(\lambda)$ alors $L'_i = L_i + \lambda L_j$
— si $E = T_{ij}$ alors $L'_i = L_j$ et $L'_j = L_i$
— si $E = D_i$ alors $L'_i = -L_i$

DÉFINITION 10. — On appelle *base* de $\mathbf{Z}^n$ le n-uplet $(\vec{\epsilon}_1, \ldots, \vec{\epsilon}_n), \vec{\epsilon}_i \in \mathbf{Z}^n$ tel que tout $x \in \mathbf{Z}^n$ se décompose de manière unique

$$x = a_1\vec{\epsilon}_1 + \cdots + a_n\vec{\epsilon}_n$$

avec $a_i \in \mathbf{Z}$

Notons $(\vec{e}_1, \ldots, \vec{e}_n)$ la base canonique de $\mathbf{Z}^n$.

THÉORÈME 6. — *$(\vec{\epsilon}_1, \ldots, \vec{\epsilon}_n)$ est une base de $\mathbf{Z}^n \Leftrightarrow$ la matrice $U = (\vec{\epsilon}_1, \ldots, \vec{\epsilon}_n)$ est unimodulaire.*

DÉFINITION 11. — Un *vecteur unimodulaire* est un vecteur de $\mathbf{Z}^n$ de coordonnées premières entre elles, c'est-à-dire de PGCD = 1.

THÉORÈME 7. — *Soit $\vec{x}$ un vecteur unimodulaire de $\mathbf{Z}^n$. Il existe une matrice unimodulaire U telle que $U \times \vec{e}_1 = \vec{x}$, c'est-à-dire telle que la première colonne de la matrice U soit le vecteur $\vec{x}$.*

La matrice unimodulaire U telle que $U \times \vec{e}_1 = \vec{x}$, $\vec{x}$ donné, est obtenue en appliquant l'algorithme du pivot dans $\mathbf{Z}^n$ décrit par BLANKINSHIP [BLA 63].

Si le vecteur $\vec{x}$ est la direction d'allocation choisie, la matrice U est telle que ses vecteurs colonnes définissent une base de $\mathbf{Z}^n$ dont le premier vecteur est $\vec{x}$. C'est à partir de cette base que nous pourons déterminer la fonction d'allocation.

Description de l'algorithme du pivot dans $\mathbf{Z}^n$.

La matrice de départ est

$$(\vec{x}, I) = \begin{pmatrix} x_1 & 1 & 0 & \ldots & 0 \\ x_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n & 0 & 0 & \ldots & 1 \end{pmatrix}$$

composée de n lignes et $n+1$ colonnes.

L'algorithme consiste à effectuer des opérations élémentaires sur les lignes de cette matrice jusqu'à obtenir une matrice dont la première colonne soit $\vec{e}_1$.

Appelons *pivot* l'un des plus petits éléments non nuls de la première colonne de la matrice.

L'algorithme s'énonce ainsi :

début
tant qu'il y a plus d'un pivot **faire**
sélectionner un pivot. Soit i son indice de ligne.
pour toutes les lignes $L_j, j \neq i$, telles que $L_{j1} \neq 0$ **faire**
déterminer q tel que $L_{j1} = q \times L_{i1} + r, 0 \leq r \leq |L_{i1}|$ (division

euclidienne)
remplacer L_j par $L_j - q \times L_i$
fin pour
fait
{soit i l'indice de ligne du pivot restant}
si $i \neq 1$ **alors** échanger les lignes L_i et L_1 **fsi**
fin

La matrice obtenue est de la forme $(\vec{e}_1, M)$, composée de n lignes et de $n+1$ colonnes, la première composante étant le vecteur $\vec{e}_1$. Les n dernières constituent la matrice carrée M.

La matrice unimodulaire U telle que $U \times \vec{e}_1 = \vec{x}$ est la matrice M^{-1}, comme le montre la vérification ci-dessous.

Vérification de l'algorithme.

La terminaison de cet algorithme a été vérifiée par BLANKINSHIP. Nous montrons seulement que :

— la première colonne de la matrice résultat est $\vec{e}_1$.

BLANKINSHIP a démontré que le pivot non nul de la matrice résultat est le plus grand commun diviseur de $x_1, \ldots, x_n$. Or, $\vec{x}$ étant unimodulaire, PGCD $(x_1, \ldots, x_n) = 1$. La première colonne de la matrice est donc $\vec{e}_1$.

— M est unimodulaire et $M^{-1} \times \vec{e}_1 = \vec{x}$.

Cet algorithme transforme la matrice $(\vec{x}, I)$ en la matrice $(\vec{e}_1, M)$. Chacune de ses étapes consiste à multiplier à gauche la matrice par une matrice élémentaire. Par conséquent, on peut écrire :

$$E_1 \ldots E_p (\vec{x}, I) = (\vec{e}_1, M)$$

Notons $E = E_1 \ldots E_p$. On a alors : $E \times \vec{x} = \vec{e}_1$ et $E \times I = M$. E étant unimodulaire, $M = E$ l'est aussi et $\vec{x} = E^{-1} \times \vec{e}_1 = M^{-1} \times \vec{e}_1$. La matrice M^{-1} est bien la matrice cherchée.

V.2. Détermination de la fonction d'allocation associée à une direction $\vec{\xi}$. — Dans la pratique, le problème s'exprime dans $\mathbf{R}^2$ ou $\mathbf{R}^3$. Nous nous situons comme précédemment dans $\mathbf{R}^3$. Les simplifications pour $\mathbf{R}^2$ sont évidentes.

Dans la base canonique $(\vec{b}_i, \vec{b}_j, \vec{b}_k)$ du domaine, le vecteur d'allocation $\vec{\xi}$ est choisi unimodulaire et tel que sa composante sur l'axe des temps

dans l'espace-temps soit non nulle, car des calculs simultanés ne peuvent s'exécuter sur la même cellule.

Lorsque le domaine D est infini, la première composante des points du domaine peut prendre une infinité de valeurs. Cette composante correspond à la coordonnée du point sur le vecteur $\vec{b_i}$ de la base canonique. Pour avoir un nombre fini de cellules, nous nous limitons, dans ce cas, à un vecteur d'allocation $\vec{\xi}$ parallèle au vecteur $\vec{b_i}$, c'est-à-dire de la forme $(\xi_1, 0, 0)_{B.C.}$.

Lorsque le sous-domaine D_2 est non vide, le réseau est constitué de deux types de cellules. Les points de D_2 doivent tous s'exécuter sur des cellules de même type. C'est pourquoi la direction d'allocation $\vec{\xi}$ doit être perpendiculaire au vecteur $\vec{b_k}$, c'est-à-dire de la forme $(\xi_1, \xi_2, 0)_{B.C.}$.

Pour la représentation initiale, la base de l'espace-temps coïncide avec la base canonique. De ce fait, dans les deux cas évoqués ci-dessus, la composante de $\vec{\xi}$ sur l'axe des temps $(O'\vec{t})$ est nulle. Ce choix pour $\vec{\xi}$ est impossible. Aucun choix de direction d'allocation n'est donc possible pour la représentation initiale dans ces deux cas.

En appliquant l'algorithme du pivot, nous pouvons déterminer une matrice unimodulaire $U = (\vec{\xi}, \vec{\xi}', \vec{\xi}'')$. D'après le théorème 6, $(\vec{\xi}, \vec{\xi}', \vec{\xi}'')$ est une base de $\mathbf{Z}^3$. On peut donc exprimer les coordonnées des points du domaine dans cette base.

Soit un point du domaine de coordonnées (i, j, k) dans la base canonique $(\vec{b_i}, \vec{b_j}, \vec{b_k})$. Ses coordonnées dans la base $(\vec{\xi}, \vec{\xi}', \vec{\xi}'')$ sont (x_1, x_2, x_3) définies par :

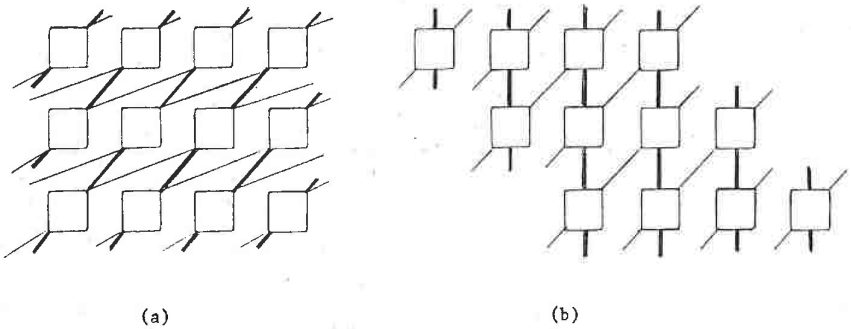
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = U^{-1} \times \begin{pmatrix} i \\ j \\ k \end{pmatrix}$$

où U est la matrice obtenue par l'algorithme du pivot.

La fonction d'allocation est donc définie par :

DÉFINITION 12. — Soit $\vec{\xi}$ une direction d'allocation et U une matrice unimodulaire constituée par trois vecteurs : $U = (\vec{\xi}, \vec{\xi}', \vec{\xi}'')$. La *fonction d'allocation*, a de $\mathbf{Z}^3$ dans $\mathbf{Z}^2$, définit pour tout point du domaine D, sur quelle cellule du réseau systolique le calcul associé à ce point doit s'exécuter. Elle est définie par :

$$\begin{array}{rccc} a: & d \in \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i, j, k) & \rightarrow & (x_2, x_3) \end{array}$$



(a) (b)

FIGURE 4.6. Configurations différentes d'un même réseau

où x_2 et x_3 sont définis par

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = U^{-1} \times \begin{pmatrix} i \\ j \\ k \end{pmatrix}$$

La matrice unimodulaire U telle que $U \times \vec{e_1} = \vec{\xi}$ n'est pas unique. Il en est donc de même pour la fonction d'allocation. Par conséquent, la configuration du réseau correspondant à une direction d'allocation $\vec{\xi}$ n'est pas unique.

Si l'on se situe dans le cadre de la réalisation d'un réseau systolique par un circuit intégré, on souhaite que la longueur des communications entre cellules soit la plus constante possible. Or, parmi différentes configurations d'un même réseau, certaines peuvent être plus intéressantes que d'autres lorsque les longueurs des différents chemins entre deux cellules sont plus homogènes. Considérons par exemple les deux configurations d'un même réseau de la figure 4.6. Appelons flot (1) les chemins de communication parallèles représentés par les traits gras, flot (2) les autres. La différence de longueur d'un chemin pour les flots (1) et (2) est plus importante pour la configuration (a) que pour la configuration (b). La configuration (b) est donc préférable car la longueur des différents chemins est plus homogène.

Notre objectif est de construire un logiciel permettant de déterminer des réseaux systoliques d'un problème donné. Nous souhaitons en particulier le

doter d'outils permettant d'obtenir différentes configurations d'un même réseau. Il convient donc de déterminer comment passer d'une configuration du réseau à une autre. Pour cela, nous utilisons le théorème suivant.

THÉORÈME 8. — *Soit U et U' deux matrices unimodulaires dont la première colonne est $\vec{\xi}$. Il existe un produit de matrices élémentaires $P = E_1 \times \cdots \times E_p$ tel que $U' = P \times U$.*

Démonstration. — Notons $U = (\vec{\xi}, \vec{\xi}_2, \vec{\xi}_3), U' = (\vec{\xi}, \vec{\xi}'_2, \vec{\xi}'_3)$.
On a $U \times \vec{e}_1 = \vec{\xi}$ et $U' \times \vec{e}_1 = \vec{\xi}$, d'où $U^{-1} \times U' \times \vec{e}_1 = U^{-1} \times \vec{\xi} = \vec{e}_1$. La première colonne de la matrice $U^{-1}U'$ est le vecteur $\vec{e}_1$.

Soit

$$U^{-1}U' = \begin{pmatrix} 1 & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{pmatrix}$$

On a donc

$$U^{-1}U' = \begin{pmatrix} 1 & a'_{12} & a'_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{pmatrix}$$

Or

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = E_{12}(a'_{12}) \times E_{13}(a'_{13})$$

Notons V la matrice

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{pmatrix}$$

Les matrices U et U' étant unimodulaires, $U^{-1}U'$ est unimodulaire et son déterminant vaut ± 1.

Or

$$\begin{aligned} det(U^{-1}U') &= det(E_{12}(a'_{12}) \times E_{13}(a'_{13}) \times V) \\ &= det(E_{12}(a'_{12})) \times det(E_{13}(a'_{13})) \times det(V) \\ &= det(V) \end{aligned}$$

car $det(E_{12}(a'_{12})) = det(E_{13}(a'_{13})) = 1$.

Donc $det(V) = \pm 1$. V est une matrice unimodulaire qui peut s'exprimer par un produit de matrices élémentaires : $V = E_1 \times \cdots \times E_q$.

On a donc $U^{-1}U' = P \Leftrightarrow U' = U \times P$ où $P = E_{12}(a'_{12}) \times E_{13}(a'_{13}) \times E_1 \times \cdots \times E_q$ avec $E_k \in \{E_{ij}(\lambda), T_{ij}, D_i, i \neq 1, j \neq 1\}, k = 1, \ldots, q$.

Appelons a la fonction d'allocation obtenue à partir de U^{-1} et a' celle obtenue à partir de U'^{-1}. La fonction a est définie par

$$a(i,j,k) = (x_2, x_3) \quad \text{avec} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = U^{-1} \times \begin{pmatrix} i \\ j \\ k \end{pmatrix}$$

De même la fonction a' est définie par

$$a'(i,j,k) = (x'_2, x'_3) \quad \text{avec} \quad \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = U'^{-1} \times \begin{pmatrix} i \\ j \\ k \end{pmatrix}$$

La matrice P étant telle que $U' = U \times P$ est un produit de matrices élémentaires noté $P = \prod_{k=1}^{l} P_k$, où chaque P_k est de la forme $E_{ij}(\lambda)$ avec $j \neq 1$, T_{ij} ou D_i avec $i \neq 1$ et $j \neq 1$, examinons l'effet de l'application de chaque type de matrice élémentaire P_k sur la matrice intermédiaire $U_0^{-1} = P_{k-1}^{-1} \times \cdots \times P_1^{-1} \times U^{-1}$ dont la fonction d'allocation associée est a_0. Notons $a_0(i,j,k) = (x_2, x_3)$.

— (1) $P_k = T_{ij}, i \neq 1, j \neq 1, i \neq j$

On peut avoir, soit $i = 2, j = 3$, soit $i = 3, j = 2$. La matrice U'_0 vérifie :

$$U'^{-1}_0 = T_{ij}^{-1} \times U_0^{-1} = T_{ij} \times U_0^{-1}$$

La matrice U'^{-1}_0 est donc obtenue à partir de U_0^{-1} en échangeant les lignes L_i et L_j. La fonction d'allocation a'_0 associée à U'^{-1}_0 peut donc s'écrire $a'_0(i,j,k) = (x_3, x_2)$.

Cela signifie que les vecteurs de base $\vec{\xi}_2$ et $\vec{\xi}_3$ ont été échangés, ce qui ne modifie pas la configuration générale du réseau, au sens où toute cellule conserve les mêmes cellules adjacentes.

— (2) $P_k = D_i, i \neq 1$

On a :

$$U'^{-1}_0 = D_i^{-1} \times U_0^{-1} = D_i \times U_0^{-1}$$

La matrice ${U'}_0^{-1}$ est donc obtenue à partir de U_0^{-1} en remplaçant la ligne L_i par $-L_i$. La fonction d'allocation a'_0 est donc définie par $a'_0(i,j,k) = (-x_2, x_3)$ pour $P_k = D_2$ et $a'_0(i,j,k) = (x_2, -x_3)$ pour $P_k = D_3$.

Le vecteur de base $\vec{\xi}_i$ est remplacé par son opposé $-\vec{\xi}_i$. Dans ce cas également, la configuration générale du réseau n'est pas modifiée.

— $(3) P_k = E_{ij}(\lambda), i \neq j, j \neq 1$

On a :

$$ {U'}_0^{-1} = E_{ij}(\lambda)^{-1} \times U_0^{-1} = E_{ij}(-\lambda) \times U_0^{-1} $$

La matrice ${U'}_0^{-1}$ est donc obtenue à partir de U_0^{-1} en remplaçant la ligne L_i par $L_i - \lambda L_j$.

Deux cas sont à distinguer.

Si $i = 1$, la fonction d'allocation a'_0 est égale à a_0. La configuration est donc inchangée.

Si $i \neq 1$, la fonction d'allocation a'_0 peut s'écrire
$a'_0(i,j,k) = (x_2 - \lambda x_3, x_3)$ pour $i = 2$,
$a'_0(i,j,k) = (x_2, x_3 - \lambda x_2)$ pour $i = 3$.

Cela signifie que le vecteur de base $\vec{\xi}_j$ est remplacé par $\vec{\xi}_j - \lambda \vec{\xi}_i$. Supposons que $i = 2$. Pour passer de la configuration dans la base $(\vec{\xi}_2, \vec{\xi}_3)$ à celle dans la base $(\vec{\xi}_2, \vec{\xi}_3 - \lambda \vec{\xi}_2)$, il faut appliquer sur le réseau une transformation T_0 qui déplace tous les points du réseau afin de faire coïncider l'image du vecteur $\vec{\xi}_3 - \lambda \vec{\xi}_2$ par T_0 avec le vecteur $\vec{\xi}_3$ de l'ancienne base. Cette transformation est définie par

$$ T_0 : (x_2, x_3) \rightarrow (x_2 - \lambda x_3, x_3) $$

C'est une transvection dont la base est la droite $(O\vec{\xi}_2)$.

En conclusion, les seules matrices élémentaires qui influencent la configuration générale du réseau sont de la forme $E_{ij}(\lambda), i \neq 1, j \neq 1$. En pratique, pour les réseaux bidimensionnels, on peut passer d'un réseau à un autre en appliquant une transvection dont la base est l'une des directions $\vec{\xi}_2$ ou $\vec{\xi}_3$ de $\mathbf{R}^2$ avec $\lambda \in \mathbf{Z}$.

V.3. Création du réseau. — Nous devons compléter la configuration des cellules du réseau obtenues à partir de la fonction d'allocation en ajoutant les chemins de communications et les données. Les familles de données sont caractérisées par leur nature :
— données constantes de cellules

— données diffusées
— données circulant systoliquement de cellules en cellules

Dans le cas où les données circulent soit par diffusion, soit systoliquement, les chemins de communications associés sont caractérisés par :
— leur orientation
— l'espacement des données sur un chemin
— le nombre de cellules de retard sur un chemin entre deux cellules de calcul consécutives
— l'ordre des données sur un chemin

Nous décrivons comment sont déterminés ces différents éléments à partir de la direction d'allocation, de la fonction d'allocation et des vecteurs générateurs associés à une famille de données. Puis, nous illustrons cela avec la détermination de deux réseaux pour le produit de convolution et du réseau hexagonal pour le produit matriciel.

Considérons,pour une représentation R donnée, une famille de données $(d_{ll'})_{l \in L, l' \in L'}$ dont le vecteur générateur est défini par

$$ \Phi_d = (\varphi_i, \varphi_j, \varphi_k)_{B.C.} = (\varphi_i, \varphi_j, \varphi_t)_{E.T.} $$

V.3.1. Nature d'une suite de données. — Le vecteur Φ_d indique qu'une donnée élémentaire $d_{ll'}$ utilisée au point $(p_1, p_2, p_3)_{B.C.}$ est aussi utilisée au point $(p_1 + \varphi_i, p_2 + \varphi_j, p_3 + \varphi_k)_{B.C.}$. Par conséquent, la donnée $d_{ll'}$ est utilisée dans les cellules référencées par $a(p_1, p_2, p_3)$ et par $a(p_1 + \varphi_i, p_2 + \varphi_j, p_3 + \varphi_k) = a(p_1, p_2, p_3) + a(\varphi_i, \varphi_j, \varphi_k) = a(p_1, p_2, p_3) + a(\Phi_d)$, où Φ_d est exprimé dans la base canonique.

Lorsque $a(\Phi_d) = (0,0)$, la donnée $d_{ll'}$ est donc utilisée dans une seule cellule. Dans ce cas, les données de la suite $(d_{ll'})_{l \in L, l' \in L'}$ sont constantes des cellules.

Lorsque $a(\Phi_d) \neq (0,0)$, la donnée $d_{ll'}$ circule donc dans le réseau, soit par diffusion, soit systoliquement. Elle est diffusée si elle est utilisée simultanément par plusieurs cellules. En raison de la définition d'un vecteur générateur dans l'espace-temps, ceci est réalisé si $\varphi_t = 0$. Par opposition, elle circule systoliquement de cellules en cellules si $\varphi_t \neq 0$.
En résumé, les données $(d_{ll'})_{l \in L, l' \in L'}$:
— sont constantes des cellules si $a(\Phi_d) = (0,0)$
— sont diffusées si $a(\Phi_d) \neq (0,0)$ et $\varphi_t = 0$
— circulent systoliquement si $a(\Phi_d) \neq (0,0)$ et $\varphi_t \neq 0$

Pour un réseau linéaire, $a(\Phi_d) \in \mathbf{Z}$, donc les conditions ci-dessus s'expriment par $a(\Phi_d) = 0$ ou $a(\Phi_d) \neq 0$.

V.3.2. Orientation d'un chemin de données. — On se situe dans le cas où les données $(d_{ll'})_{l \in L, l' \in L'}$ circulent, soit avec diffusion, soit systoliquement, c'est-à-dire où $a(\Phi_d) \neq (0,0)$. Lorsqu'une donnée $d_{ll'}$ circule, elle est uitlisée dans les cellules $C_{a(p_1,p_2,p_3)}$ et $C_{a(p_1,p_2,p_3)+a(\Phi_d)}$.

a. Cas des données diffusées.

Pour un réseau linéaire, la diffusion est effectuée sur toutes les cellules adjacentes du réseau.

Pour un réseau bidimensionnel, la diffusion peut être horizontale, verticale ou oblique. L'orientation de cette diffusion est donnée par $a(\Phi_d)$. En effet, supposons la fonction d'allocation a définie par :

$$\begin{array}{rccc} a: & D \subset \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k)_{E.C.} & \rightarrow & (x,y) \end{array}$$

où x et y représentent les coordonnées de la cellule associée dans le plan où est représenté le réseau, muni des axes $(O\vec{x})$ et $(O\vec{y})$. Notons $a(\Phi_d) = (x_0, y_0)$.

— Si $x_0 \neq 0$ et $y_0 = 0$ alors les données sont utilisées dans les cellules $C_{a(p_1,p_2,p_3)}$ et $C_{a(p_1,p_2,p_3)+(x_0,0)}$. La diffusion est donc parallèle à l'axe $(O\vec{x})$.

— Si $x_0 = 0$ et $y_0 \neq 0$ alors les données sont utilisées dans les cellules $C_{a(p_1,p_2,p_3)}$ et $C_{a(p_1,p_2,p_3)+(0,y_0)}$. La diffusion est donc parallèle à l'axe $(O\vec{y})$.

— Si $x_0 \neq 0$ et $y_0 \neq 0$ alors la diffusion est oblique par rapport aux axes $(O\vec{x})$ et $(O\vec{y})$. Dans la version actuelle du logiciel SYSTOL, nous avons choisi de ne pas visualiser des réseaux avec de telles diffusions car les chemins de communication "chevauchent" des cellules.

b. Cas des données à circulation systolique.

On est dans le cas où $a(\Phi_d) \neq (0,0)$ et $\varphi_t \neq 0$. Considérons une donnée $d_{ll'}$ utilisée en un point $(i,j,k)_{E.C.} = (i,j,t)_{E.T.}$ donné, sur la cellule $C_{a(i,j,k)}$, à l'instant t. Cette donnée est également utilisée au point $(i+\varphi_i, j+\varphi_j, t+\varphi_t)_{E.T.}$ sur la cellule $C_{a(i,j,k)+a(\Phi_d)}$, à l'instant $t+\varphi_t$. L'orientation (ou sens) du flot dépend donc du signe de φ_t :

— si $\varphi_t > 0$ alors le flot est orienté de la cellule $C_{a(i,j,k)}$ vers la cellule $C_{a(i,j,k)+a(\Phi_d)}$.

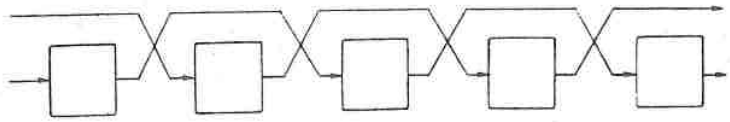
— si $\varphi_t < 0$ alors le flot est orienté de la cellule $C_{a(i,j,k)+a(\Phi_d)}$ vers la cellule $C_{a(i,j,k)}$.

Pour les réseaux linéaires, les cellules sont définies par un indice : $a(i,k) \in \mathbf{Z}$. Si l'on suppose les cellules indicées par ordre croissant de gauche à droite, le sens du flot est :

— de la gauche vers la droite si $\varphi_t > 0$ et $a(\Phi_d) > 0$ ou si $\varphi_t < 0$ et $a(\Phi_d) < 0$.

— de la droite vers la gauche si $\varphi_t > 0$ et $a(\Phi_d) < 0$ ou si $\varphi_t < 0$ et $a(\Phi_d) > 0$.

Notons que lorsque $|a(\Phi_d)| > 1$, le flot ne passe pas à travers toutes les cellules du réseau. Il "saute" $|a(\Phi_d)| - 1$ cellules. Ainsi, pour une suite de données caractérisée par $\varphi_t > 0$ et $a(\Phi_d) = 2$, le flot correspondant est de la forme :



Dans la version actuelle du logiciel SYSTOL, nous nous limitons aux tracé des réseaux pour lesquels $|a(\Phi_d)| = 1$. Il ne permet donc pas de déterminer un réseau de la forme ci-dessus.

Pour les réseaux bidimensionnels, les cellules sont définies par deux indices : $a(i,j,k) \in \mathbf{Z}^2$. Notons $a(\Phi_d) = (x_0, y_0)$ et supposons le réseau représenté dans le plan muni de deux axes $(O\vec{x})$ et $(O\vec{y})$ correspondant respectivement au premier et au deuxième indice de numérotation des cellules.

— Le flot est parallèle à l'axe $(O\vec{x})$ si $x_0 \neq 0$ et $y_0 = 0$. De plus, pour qu'il ne "saute" pas de cellules il faut que $|x_0| = 1$.

— Le flot est parallèle à l'axe $(O\vec{y})$ si $x_0 = 0$ et $y_0 \neq 0$. De plus, pour qu'il ne "saute" pas de cellules il faut que $|y_0| = 1$.

— Le flot est oblique par rapport aux axes $(O\vec{x})$ et $(O\vec{y})$ si $x_0 \neq 0$ et $y_0 \neq 0$. De plus, pour qu'il ne saute pas de cellules, il faut que $|x_0| = 1$ ou $|y_0| = 1$.

Dans sa version actuelle, SYSTOL ne permet de représenter que des flots qui ne "sautent" pas de cellules.

V.3.3. Espacement des données sur les flots. — Soit $\vec{\xi}$ la direction d'allocation. Ses coordonnées dans l'espace-temps sont $(\xi_i, \xi_j, \xi_t)_{E.T.}$ avec $\xi_t \neq 0$ et PGCD $(\xi_i, \xi_j, \xi_t) = \pm 1$ par définition des choix possibles pour $\vec{\xi}$.

Considérons les données qui circulent systoliquement. Soit un point de calcul $(i, j, t)_{E.T.}$ dans l'espace-temps. Ce calcul s'effectue sur une cellule définie par la fonction a. Les calculs exécutés sur cette cellule sont définis par $(i + \nu\xi_i, j + \nu\xi_j, t + \nu\xi_t)_{E.T.} \in D, \nu \in \mathbf{Z}$. L'abscisse temporelle de ces calculs, c'est-à-dire l'instant auxquels ils sont exécutés, est $t + \nu\xi_t$. Donc $|\xi_t|$ intervalles de temps séparent deux calculs consécutifs sur une même cellule. Deux données utilisées pour deux calculs consécutifs sur une cellule doivent être disponibles sur le flot d'entrée correspondant de cette cellule aux instants t et $t + |\xi_t|$. Aux instants intermédiaires t_0, $t < t_0 < t + |\xi_t|$, aucune donnée effective ne doit être disponible sur le flot. Deux données effectives consécutives sur le flot considéré doivent donc être séparées par $|\xi_t| - 1$ données fictives (ou espaces).

L'espacement des données sur les flots est défini par $|\xi_t| - 1$. Si $|\xi_t| = 1$, la cellule est active à chaque intervalle de temps. Les données sont donc contigues (sans espacement) sur le flot. Notons que l'espacement est le même pour tous les flots de données à circulation systolique.

Pour les flots diffusés, dans les réseaux linéaires associés à des représentations valides, il n'y a pas d'espacement entre les données. En effet, si la représentation considérée est telle qu'il y a au moins un calcul à tout instant donné, alors les données sont contigues sur le flot. Si par contre, la représentation considérée est telle que τ intervalles de temps, $\tau > 1$, séparent deux calculs consécutifs, alors il existe une représentation qui la précède pour laquelle les réseaux ont déjà été déterminés. La représentation considérée n'est donc pas valide, c'est pourquoi on ne cherche pas les réseaux qui lui sont associés.

Pour les flots diffusés, dans les réseaux bidimensionnels, l'existence d'espacements entre les données dépend de la manière dont sont actives les cellules associées à un flot de diffusion. Si toutes ces cellules sont actives aux mêmes instants, il y a $|\xi_t| - 1$ espaces entre deux données diffusées du flot considéré. Si par contre les cellules associées à un flot de diffusion, sont actives à des instants successifs, il n'y a pas d'espace entre les données diffusées.

V.3.4. Nombre de cellules de retard sur les flots. — Tous les flots de données ne circulent pas à la même vitesse. Pour retarder un flot, des cellules de retard sont introduites sur le flot entre deux cellules de calcul adjacentes.

Des cellules de retard ne peuvent apparaitre que sur des flots à circulation systolique. On est donc dans le cas où $a(\Phi_d) \neq 0$ et $\varphi_t \neq 0$ pour une famille de données $(d_{ll'})_{l \in L, l' \in L'}$ quelconque.

Soit un point de calcul $(i, j, t)_{E.T.}$ dans l'espace-temps utilisant une donnée $d_{ll'}$. Les points utilisant cette donnée sont définis par $(i + \nu\varphi_i, j + \nu\varphi_j, t + \nu\varphi_t)_{E.T.} \in D, \nu \in \mathbf{Z}$. Leur abscisse temporelle est $t + \nu\varphi_t$. Donc $|\varphi_t|$ intervalles de temps séparent deux utilisations successives de cette donnée, utilisée en deux cellules consécutives sur le flot associé. Cette donnée, disponible sur le flot associé, en entrée d'une cellule à un instant t donné, doit donc être disponible en entrée de la cellule consécutive $|\varphi_t|$ intervalles de temps plus tard. Elle doit donc transiter par $|\varphi_t| - 1$ cellules de retard situées entre les deux cellules consécutives sur le flot.

Le nombre de cellules de retard sur le flot associé à la suite de données $(d_{ll'})_{l \in L, l' \in L'}$ est donc de $|\varphi_t| - 1$ où φ_t est défini par $\Phi_d = (\varphi_i, \varphi_j, \varphi_t)_{E.T.}$.

V.3.5. Ordre des données sur les flots. — Il est déterminé à partir de la fonction d'allocation et de la spécification systolique du problème, en particulier des fonctions h_d définissant comment sont utilisées les données du problème aux points du domaine.

Il nécessite la mise en œuvre d'un algorithme qui considère successivement chacun des cas. Nous n'exposons pas ici cet algorithme qui est réalisé dans le logiciel SYSTOL.

V.4. Exemples de détermination de réseaux. — Nous utilisons les résultats énoncés pour déterminer les réseaux systoliques associés au produit de convolution présentés aux figures 2-4 et 2-5, ainsi que le réseau hexagonal associé au produit matriciel (*cf.* figure 2-11).

V.4.1. Produit de convolution. — Le produit de convolution est caractérisé par le domaine

$$D = \{(i, k) \in \mathbf{Z}^2 \quad | \quad 0 \leq i \leq n - m, 0 \leq k \leq m, n = 7, m = 2\}$$

et par trois familles de données $(w_k)_{k=0,\ldots,2}$, $(x_i)_{i=0,\ldots,7}$ et $(y_i)_{i=0,\ldots,5}$ dont les vecteurs générateurs sont $\Phi_w = (1, 0)_{B.C.}$, $\Phi_x = (1, -1)_{B.C.}$ et

$\Phi_y = (0,1)_{B.C.}$. La liste des données utilisées en un point de D est définie par $h_w(i,k) = k$, $h_x(i,k) = i+k$ et $h_y(i,k) = i$.

EXEMPLE 1. — Considérons la représentation R du produit de convolution définie par $R = (T_d(O'\vec{i}) \circ T_c(O'\vec{i}))_{R_0}(R_0)$ où R_0 est la représentation initiale. Elle est visualisée figure 3-10(b) et caractérisée par

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Choisissons comme direction d'allocation $\vec{\xi} = (1,0)_{B.C.} = (1,1)_{E.T.}$. La fonction d'allocation est déterminée en utilisant l'algorithme du pivot. Pour cet algorithme, la matrice de départ est $(\vec{\xi}, I) = \left(\begin{smallmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$. Elle est constituée d'un seul pivot situé sur la première ligne. L'algorithme ne modifie donc par cette matrice. La fonction d'allocation est donc définie par :

$$\begin{array}{rccc} a: & D \subset \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,k) & \rightarrow & x_2 \end{array}$$

où x_2 est défini par $\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)\left(\begin{smallmatrix} i \\ k \end{smallmatrix}\right) = \left(\begin{smallmatrix} i \\ k \end{smallmatrix}\right)$. On a donc

$$\forall (i,k) \in D, a(i,k) = k$$

Le réseau linéaire est constitué de 3 cellules car $a(i,k) = k$ et $0 \leq k \leq 2$. On numérote les cellules dans l'ordre croissant de gauche à droite.

L'espacement des données sur les flots de circulation systolique est donné par $|\xi_t| - 1 = 0$. Il n'y a donc pas d'espaces entre deux données consécutives sur un flot.

Considérons successivement chaque famille de données et déterminons leur position dans le réseau.

— la suite $(w_k)_{k=0,\ldots,2}$ est caractérisée par $\Phi_w = (1,0)_{B.C.}$. Les données sont constantes des cellules car $a(\Phi_w) = a(1,0) = 0$.

De plus, un point quelconque (i,k) du domaine D utilise la donnée $w_{h_w(i,k)} = w_k$ de la suite et s'exécute sur la cellule $C_{a(i,k)} = C_k$. Donc la k-ième donnée w_k est constante de la k-ième cellule C_k.

— la suite $(x_i)_{i=0,\ldots,7}$ est caractérisée par $\Phi_x = (1,-1)_{B.C.} = (1,-1)_{E.T.}$. Les données x_i circulent systoliquement dans le réseau car $a(\Phi_x) = a(1,-1) = -1 \neq 0$ et $\varphi_t \neq 0$.

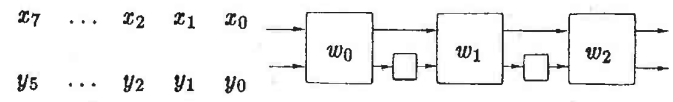


FIGURE 4.7. Réseau systolique correspondant à l'exemple 1

De plus, on a $\varphi_t < 0$. Le flot est donc orienté de la cellule $C_{p+a(\Phi_x)} = C_{p-1}$ vers la cellule C_p, c'est-dire de gauche à droite dans le réseau linéaire.

Il n'y a pas de cellules de retard sur le flot associé car $|\varphi_t| - 1 = 0$.

Un point quelconque (i,k) du domaine D utilise la donnée $x_{h_x(i,k)} = x_{i+k}$ sur la cellule $C_{a(i,k)} = C_k$ à l'instant $t = i+2k$, tandis que le point $(i+1,k)$ utilise la donnée $x_{h_x(i+1,k)} = x_{i+k+1}$ sur la cellule $C_{a(i+1,k)} = C_k$ à l'instant $t' = i+1+2k = t+1$. Une cellule utilise donc successivement les données x_l et $x_{l+1}, l = 0,\ldots,6$. La suite $(x_i)_{i=0,\ldots,7}$ est donc organisée sur le flot associé dans l'ordre croissant de ses indices.

— la suite $(y_i)_{i=0,\ldots,5}$ est caractérisée par $\Phi_y = (0,1)_{B.C.} = (0,2)_{E.T.}$. Les données y_i circulent systoliquement dans le réseau car $a(\Phi_y) = a(0,1) = 1 \neq 0$ et $\varphi_t \neq 0$.

De plus, on a $\varphi_t > 0$. Le flot est donc orienté de la cellule C_p vers la cellule $C_{p+a(\Phi_y)} = C_{p+1}$, c'est-dire de gauche à droite dans le réseau linéaire.

Il y a une cellule de retard sur le flot associé car $|\varphi_t| - 1 = 1$.

Un point quelconque (i,k) du domaine D utilise la donnée $y_{h_y(i,k)} = y_i$ sur la cellule $C_{a(i,k)} = C_k$ à l'instant $t = i+2k$, tandis que le point $(i+1,k)$ utilise la donnée $y_{h_y(i+1,k)} = y_{i+1}$ sur la cellule $C_{a(i+1,k)} = C_k$ à l'instant $t' = i+1+2k = t+1$. Une cellule utilise donc successivement les données y_l et $y_{l+1}, l = 0,\ldots,4$. La suite $(y_i)_{i=0,\ldots,5}$ est donc organisée sur le flot associé dans l'ordre croissant de ses indices.

Le réseau ainsi obtenu est visualisé figure 4.7.

EXEMPLE 2. — Considérons la représentation R du produit de convolution visualisée figure 3-8 (a) et définie par $R = T_c(O'\vec{i})_{R_0}(R_0)$ où R_0 est

la représentation initiale. Elle est caractérisée par

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Choisissons comme direction d'allocation $\vec{\xi} = (1,1)_{B.C.} = (1,2)_{E.T.}$. Appliquons l'algorithme du pivot pour déterminer la fonction d'allocation. Les différentes étapes de l'algorithme transforment la matrice ainsi :

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$$

La fonction d'allocation est donc définie par :

$$\begin{array}{rccc} a: & D \subset \mathbf{Z}^2 & \rightarrow & \mathbf{Z} \\ & (i,k) & \rightarrow & x_2 \end{array}$$

où x_2 est défini par $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} i \\ k \end{pmatrix} = \begin{pmatrix} i \\ -i+k \end{pmatrix}$. On a donc

$$\forall (i,k) \in D, a(i,k) = k - i$$

Le réseau linéaire est constitué de 8 cellules $C_{-5}, C_{-4}, \ldots, C_0, C_1, C_2$ car $a(i,k) = k - i$ et $0 \leq i \leq 5$, $0 \leq k \leq 2$. On numérote les cellules dans l'ordre croissant de gauche à droite.

L'espacement des données sur les flots de circulation systolique est donné par $|\xi_t| - 1 = 1$. Il y a donc un espace entre deux données consécutives sur un flot.

Considérons successivement chaque famille de données et déterminons leur position dans le réseau.

— la suite $(x_i)_{i=0,\ldots,7}$ est caractérisée par $\Phi_x = (1,-1)_{B.C.} = (1,0)_{E.T.}$. Les données x_i sont diffusées car $a(\Phi_x) = a(1,-1) = -2 \neq 0$ et $\varphi_t = 0$.

De plus, un point quelconque (i,k) de D utilise la donnée $x_{h_x(i,k)} = x_{i+k}$ sur la cellule $C_{a(i,k)} = C_{k-i}$ à l'instant $t = i + k$, tandis que le point $(i+1, k+1)$ utilise la donnée $x_{h_x(i,k)} = x_{i+k+2}$ sur la cellule $C_{a(i+1,k+1)} = C_{k-i}$ à l'instant $t' = i + k + 2$. Une cellule utilise donc successivement x_t et x_{t+2}. La suite $(x_i)_{i=0,\ldots,7}$ est donc diffusée dans l'ordre croissant de ses indices.

— la suite $(w_k)_{k=0,\ldots,2}$ est caractérisée par $\Phi_w = (1,0)_{B.C.} = (1,1)_{E.T.}$. Les données w_k circulent systoliquement dans le réseau car $a(\Phi_w) = a(1,0) = -1 \neq 0$ et $\varphi_t \neq 0$.

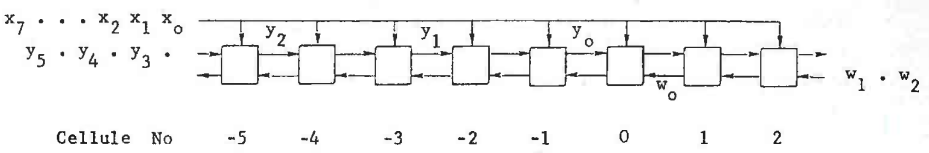


FIGURE 4.8. Réseau systolique correspondant à l'exemple 2

De plus, on a $\varphi_t > 0$. Le flot est donc orienté de la cellule C_p vers la cellule $C_{p+a(\Phi_w)} = C_{p-1}$, c'est-dire de droite à gauche dans le réseau.

Il n'y a pas de cellules de retard sur le flot associé car $|\varphi_t| - 1 = 0$.

Un point quelconque (i,k) de D utilise la donnée $w_{h_w(i,k)} = w_k$ sur la cellule $C_{a(i,k)} = C_{k-i}$ à l'instant $t = i+k$, tandis que le point $(i+1, k+1)$ utilise la donnée $w_{h_w(i+1,k+1)} = w_{k+1}$ sur la cellule $C_{a(i+1,k+1)} = C_{k-i}$ à l'instant $t' = i + k + 2 = t + 2$. Une cellule utilise donc successivement les données w_k et w_{k+1}. La suite $(w_k)_{k=0,\ldots,2}$ est donc organisée sur le flot associé dans l'ordre croissant de ses indices. De plus, à l'instant $t = 0$, la donnée w_0 est utilisée en entrée de la cellule $C_{a(0,0)} = C_0$.

— la suite $(y_i)_{i=0,\ldots,5}$ est caractérisée par $\Phi_y = (0,1)_{B.C.} = (0,1)_{E.T.}$. Les données y_i circulent systoliquement dans le réseau car $a(\Phi_y) = a(0,1) = 1 \neq 0$ et $\varphi_t \neq 0$.

De plus, on a $\varphi_t > 0$. Le flot est donc orienté de la cellule C_p vers la cellule $C_{p+a(\Phi_y)} = C_{p+1}$, c'est-dire de gauche à droite dans le réseau linéaire.

Il n' y a pas de cellule de retard sur le flot associé car $|\varphi_t| - 1 = 0$.

Un point quelconque (i,k) du domaine D utilise la donnée $y_{h_y(i,k)} = y_i$ sur la cellule $C_{a(i,k)} = C_{k-i}$ à l'instant $t = i + k$, tandis que le point $(i+1, k+1)$ utilise la donnée $y_{h_y(i+1,k+1)} = y_{i+1}$ sur la cellule $C_{a(i+1,k+1)} = C_{k-i}$ à l'instant $t' = i + k + 2 = t + 2$. Une cellule utilise donc successivement les données y_i et y_{i+1}. La suite $(y_i)_{i=0,\ldots,5}$ circule donc systoliquement dans l'ordre croissant de ses indices. De plus, à l'instant $t = 0$, la donnée y_0 est utilisée en entrée de la cellule $C_{a(0,0)} = C_0$.

Le réseau ainsi obtenu est visualisé figure 4.8.

V.4.2. Le produit matriciel. — Ce problème est caractérisé par le domaine

$$D = \{(i,j,k) \in \mathbf{Z}^3 \quad | \quad 1 \leq i \leq m, 1 \leq j \leq p, 1 \leq k \leq n\}$$

avec $m = 2, n = 2, p = 3$ et par trois familles de données

$$(a_{ij})_{i=1,\ldots,m,j=1,\ldots,n}, (b_{ij})_{i=1,\ldots,n,j=1,\ldots,p}, (c_{ij})_{i=1,\ldots,m,j=1,\ldots,p}$$

dont les vecteurs générateurs associés sont

$$\Phi_a = (0,1,0), \Phi_b = (1,0,0), \Phi_c = (0,0,1)$$

La liste des données utilisées en un point de D est définie par

$$h_a(i,j,k) = (i,k), h_b(i,j,k) = (k,j), h_c(i,j,k) = (i,j)$$

Considérons la représentation R définie par $R = (T_c(O'\vec{i}) \circ T_c(O'\vec{j}))_{R_0}(R_0)$. Elle est caractérisée par

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Choisissons comme direction d'allocation $\vec{\xi} = (1,1,1)_{B.C.} = (1,1,3)_{E.T.}$. La fonction d'allocation est déterminée par l'algorithme du pivot qui effectue les transformations suivantes :

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

La fonction d'allocation est donc définie par

$$\begin{array}{rccc} a: & D \subset \mathbf{Z}^3 & \rightarrow & \mathbf{Z}^2 \\ & (i,j,k) & \rightarrow & (x_2, x_3) \end{array}$$

avec

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} i \\ j \\ k \end{pmatrix} = \begin{pmatrix} i \\ -i+j \\ -i+k \end{pmatrix}$$

donc $\forall (i,j,k) \in D, a(i,j,k) = (j-i, k-i)$

Le réseau est bidimensionnel. La numérotation d'une cellule est donnée par $(j-i, k-i)$ où $1 \leq i \leq 2, 1 \leq j \leq 3, 1 \leq k \leq 2$. Supposons le plan muni d'un repère $(O\vec{x}, O\vec{y})$. L'axe $(O\vec{x})$ correspond à la première composante d'une cellule : pour un point (i,j,k) de D, cette composante est $j-i$. L'axe $(O\vec{y})$ correspond à la deuxième composante d'une cellule : $k-i$ pour un point (i,j,k) de D.

Appliquée à l'ensemble des points du domaine, la fonction d'allocation prend dix valeurs correspondant donc à dix cellules représentées dans le plan $(O\vec{x}, O\vec{y})$ sur la figure 4.9, la cellule supérieure gauche étant la cellule de coordonnées $(-1,-1)$.

L'espacement des données sur les flots de circulation systolique est donné par $|\xi_t| - 1 = 2$. Il y a donc 2 espaces entre les données consécutives sur un flot.

Considérons successivement chacune des familles de données et déterminons leur position dans le réseau.

— la suite $(a_{ij})_{i=1,\ldots,m,j=1,\ldots,n}$ est caractérisée par $\Phi_a = (0,1,0)_{B.C.} = (0,1,1)_{E.T.}$.

Les données circulent systoliquement dans le réseau car $a(\Phi_a) = a(0,1,0) = (1,0) \neq (0,0)$ et $\varphi_t \neq 0$.

De plus, on a $\varphi_t > 0$. Le flot est donc orienté d'une cellule $C_{(p,q)}$ vers la cellule $C_{(p,q)+a(\Phi_a)} = C_{(p+1,q)}$, c'est-à-dire dans le sens de l'axe $(O\vec{x})$.

Il n'y a pas de cellules de retard sur le flot associé car $|\varphi_t| - 1 = 0$.

Un point (i,j,k) de D utilise la donnée $a_{h_a(i,j,k)} = a_{ik}$ sur la cellule $C_{a(i,j,k)} = C_{j-i,k-i}$ à l'instant $t = i+j+k$, tandis que le point défini par $(i+1, j+1, k+1)$ utilise la donnée $a_{h_a(i+1,j+1,k+1)} = a_{i+1,k+1}$ sur la cellule $C_{a(i+1,j+1,k+1)} = C_{j-i,k-i}$ à l'instant $t = i+j+k+3$.

Une cellule utilise donc successivement les données a_{ik} et $a_{i+1,k+1}$. La suite $(a_{ij})_{i=1,\ldots,2,j=1,\ldots,2}$ est donc organisée sur les flots parallèles à l'axe $(O\vec{x})$ par diagonales dans l'ordre croissant des indices.

— la suite $(b_{ij})_{i=1,\ldots,n,j=1,\ldots,p}$ est caractérisée par $\Phi_b = (1,0,0)_{B.C.} = (1,0,1)_{E.T.}$.

Les données circulent systoliquement dans le réseau car $a(\Phi_b) = a(1,0,0) = (-1,-1) \neq (0,0)$ et $\varphi_t \neq 0$.

De plus, on a $\varphi_t > 0$. Le flot est donc orienté d'une cellule $C_{(p,q)}$ vers la cellule $C_{(p,q)+a(\Phi_a)} = C_{(p-1,q-1)}$, c'est-à-dire parallèlement à la première diagonale dans le repère $(O\vec{x}, O\vec{y})$ de direction $(-1,-1)$.

Il n'y a pas de cellules de retard sur le flot associé car $|\varphi_t| - 1 = 0$.

Un point (i,j,k) de D utilise la donnée $b_{h_b(i,j,k)} = b_{kj}$ sur la cellule $C_{a(i,j,k)} = C_{j-i,k-i}$ à l'instant $t = i+j+k$, tandis que le point défini par $(i+1,j+1,k+1)$ utilise la donnée $b_{h_b(i+1,j+1,k+1)} = b_{k+1,j+1}$ sur la cellule $C_{a(i+1,j+1,k+1)} = C_{j-i,k-i}$ à l'instant $t = i+j+k+3$.

Une cellule utilise donc successivement les données b_{kj} et $b_{k+1,j+1}$. La suite $(b_{kj})_{k=1,\ldots,2,j=1,\ldots,3}$ est donc organisée par diagonale dans l'ordre croissant de ces indices dans la direction de la première diagonale du repère $(O\vec{x}, O\vec{y})$.

— la suite $(c_{ij})_{i=1,\ldots,m,j=1,\ldots,p}$ est caractérisée par $\Phi_c = (0,0,1)_{B.C.} = (0,0,1)_{E.T.}$.

Les données circulent systoliquement dans le réseau car $a(\Phi_c) = a(0,0,1) = (0,1) \neq (0,0)$ et $\varphi_t \neq 0$.

De plus, on a $\varphi_t > 0$. Le flot est donc orienté d'une cellule $C_{(p,q)}$ vers la cellule $C_{(p,q)+a(\Phi_a)} = C_{(p,q+1)}$, c'est-à-dire dans le sens de l'axe $(O\vec{y})$.

Il n'y a pas de cellules de retard sur le flot associé car $|\varphi_t| - 1 = 0$.

Un point (i,j,k) de D utilise la donnée $c_{h_c(i,j,k)} = c_{ij}$ sur la cellule $C_{a(i,j,k)} = C_{j-i,k-i}$ à l'instant $t = i+j+k$, tandis que le point défini par $(i+1,j+1,k+1)$ utilise la donnée $c_{h_c(i+1,j+1,k+1)} = c_{i+1,j+1}$ sur la cellule $C_{a(i+1,j+1,k+1)} = C_{j-i,k-i}$ à l'instant $t = i+j+k+3$.

Une cellule utilise donc successivement les données c_{ij} et $c_{i+1,j+1}$. La suite $(c_{ij})_{i=1,\ldots,2,j=1,\ldots,3}$ est donc organisée sur les flots parallèles à l'axe $(O\vec{y})$ par diagonales dans l'ordre croissant des indices.

Le réseau ainsi décrit est représenté figure 4.9. Si on applique sur le réseau une transvection parallèle à l'axe $(O\vec{y})$ de longueur -1, on obtient une autre configuration du réseau hexagonal, représentée figure 4.10.

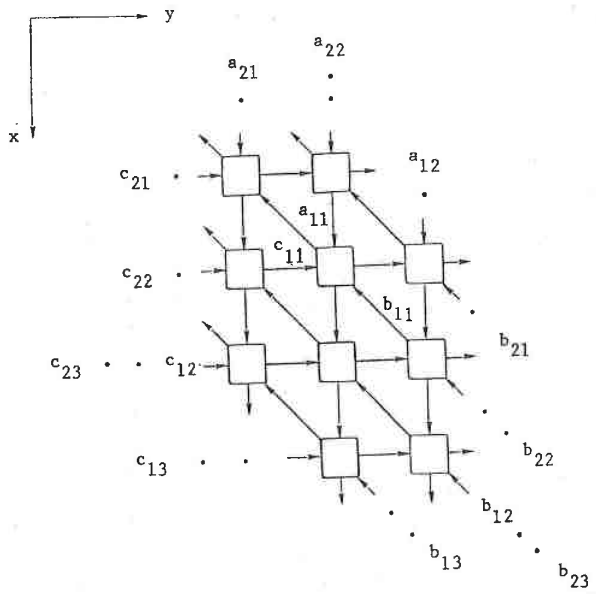


Figure 4.9. Une configuration du réseau hexagonal

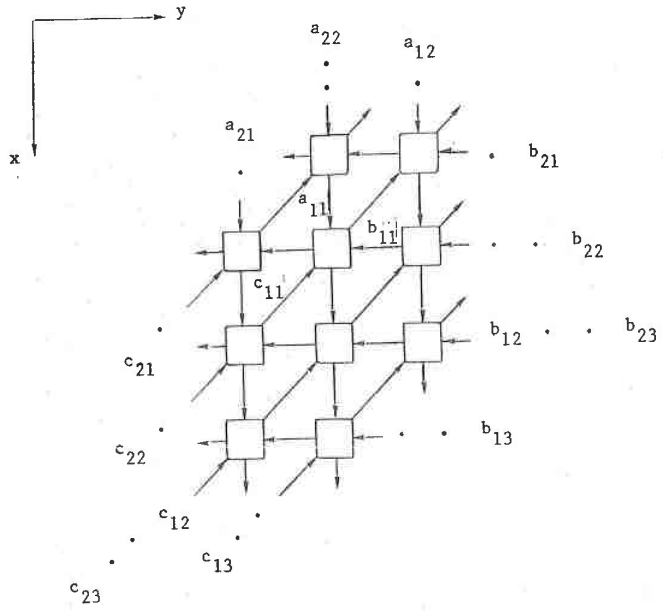


Figure 4.10. Une autre configuration du réseau hexagonal

CHAPITRE 5

PRÉSENTATION DU LOGICIEL SYSTOL

La méthode de conception d'algorithmes systoliques proposée est mise en œuvre dans un logiciel appelé SYSTOL qui détermine, pour un problème donné, défini par sa spécification systolique, un ensemble de réseaux systoliques solution de ce problème. Ce logiciel a été réalisé en Pascal sur SM 90 avec visualisation et simulation d'exécution des réseaux obtenus sur un écran bitmap Numelec.

Nous décrivons ses différentes fonctions, la manière dont il est organisé ainsi que ses limites actuelles et ses extensions futures et nous présentons un exemple d'utilisation du logiciel avec le problème du produit de convolution. Quelques unes des procédures constituant le logiciel sont présentées en annexe.

I. Les fonctions du logiciel

I.1. Saisie conversationnelle de la spécification systolique.

L'utilisateur déduit, du système d'équations récurrentes caractérisant le problème, sa spécification systolique. Cette déduction est immédiate ainsi que nous l'avons exposé au chapitre 4. Elle n'est qu'une codification de ce système et sert d'entrée dans le logiciel. L'utilisateur doit définir :

— les bornes du domaine,

— le nom et la nature (donnée simple, suite simple ou double) de chaque famille de données,

— pour chaque suite de données, l'indice (ou les indices) de l'élément de la suite utilisé en un point quelconque du domaine de coordonnées (i,k) (resp. (i,j,k)) pour un problème dans $\mathbf{R}^2$ (resp. $\mathbf{R}^3$). Un tel indice est défini par une fonction h de i,k ou i,j,k. Cette fonction doit actuellement être entrée sous forme d'une expression postfixée.

— la fonction de calcul réalisée en un point quelconque du domaine. Ses arguments sont les noms des différentes familles de données du problème. Elle est également entrée sous forme d'une expression postfixée.

I.2. Calcul de l'ensemble des représentations.

Cette phase de travail est totalement automatique et transparente à l'utilisateur.

Après avoir calculé le vecteur générateur associé à chaque famille de données du problème, le programme les utilise pour déterminer les nouvelles représentations par application de transformations sous certaines conditions portant sur les valeurs des vecteurs générateurs (*cf.* chapitre 4).

Il construit une arborescence des représentations. Chaque nœud correspond à une représentation, la représentation initiale étant associée à la racine. Chaque arc correspond à une transformation appliquée du nœud précédent et qui donne pour image la représentation associée au nœud suivant.

Pour chaque représentation, caractérisée par les valeurs de b_1, b_2, b_3 et v (*cf.* chapitre 4), il mémorise si elle est valide et si elle précède ou non une autre représentation déjà obtenue.

Dans la suite, chaque représentation R est considérée successivement par un parcours de cette arborescence. Elle est définie par la suite de transformations S, telle que $R = S_{R_0}(R_0)$, qui est affichée à l'écran. A la fin du parcours, l'utilisateur peut en demander un nouveau.

I.3. Choix d'une direction d'allocation.

Pour chaque représentation, l'utilisateur peut choisir par un dialogue interactif un ensemble de directions d'allocation, définies dans la base canonique du problème. Le logiciel teste si la direction proposée est compatible avec la représentation considérée, c'est-à-dire si son abscisse temporelle est non nulle.

Notons que pour un problème caractérisé par un domaine infini (pas de borne supérieure à l'indice i du domaine), le choix n'est pas possible. L'unique direction d'allocation autorisée est imposée par le logiciel : $\vec{\xi} = (1,0,0)_{B.C.}$. Cela permet de garantir que le réseau obtenu est constitué d'un nombre fini de cellules.

I.4. Calcul des caractéristiques d'un réseau.

Cette phase de travail est également automatique. Pour une représentation et une direction d'allocation choisie, le logiciel calcule la fonction d'allocation et détermine les caractéristiques du réseau, selon le processus décrit au chapitre 4 :
— le nombre et la disposition des cellules,
— la nature des familles de données (données constantes de cellules, données diffusées, données circulant entre les cellules),
— l'orientation des flots (pour la diffusion ou la circulation systolique),
— l'espacement des données sur les flots,
— le nombre de cellules de retard sur les différents flots à circulation systolique,
— l'ordre des données sur les flots.

I.5. Dessin et simulation d'exécution des réseaux.

Pour une représentation et une direction d'allocation données, le logiciel dessine le réseau au fur et à mesure du calcul de ses caractéristiques.

Une cellule est représentée par une boite, les données sont dessinées :
— dans les boites lorsque elles sont constantes des cellules (le flot de sortie pointillé supplémentaire, nécessaire quand le résultat est constante d'une cellule, n'est pas représenté),
— sur les flèches correspondants aux flots lorsqu'elles circulent,
— devant le flot correspondant lorsqu'elles sont diffusées.

La simulation d'exécution est réalisée au pas à pas à la demande de l'utilisateur en déplaçant les données sur les flots.

L'utilisateur a également la possibilité de mémoriser l'image point par point d'un réseau afin de pouvoir en faire une copie sur papier, en utilisant un programme de "hard-copy" disponible sur le système.

II. Organisation du logiciel

Ce logiciel est écrit en Pascal sur SM 90. Il est organisé en modules. L'ensemble des sources représentent environ 4800 lignes, soit 320 blocs

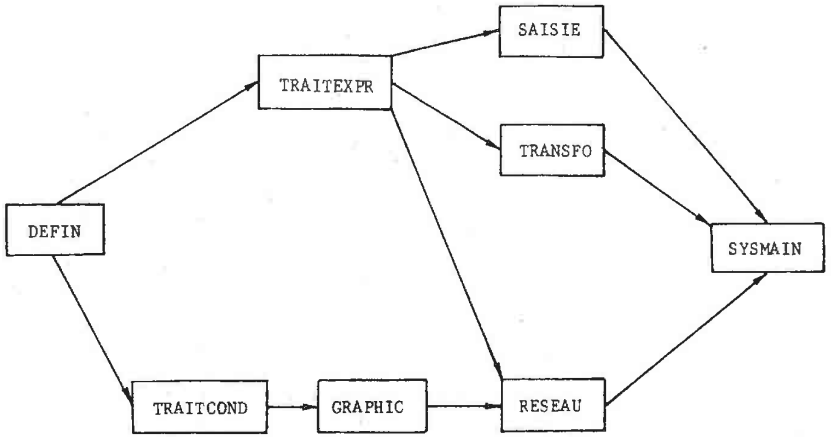


FIGURE 5.1. Organisation modulaire de SYSTOL

Son schéma général d'organisation est représenté figure 5.1 sur lequel apparaissent des flèches symbolisant les dépendances entre les modules.

Décrivons succinctement les différents modules :

— le module DEFIN définit l'ensemble des types et des variables Pascal correspondants à la spécification systolique du problème et à l'arbre des représentations.

— le module SAISIE lit l'ensemble des informations définissant la spécification systolique du problème.

— le module TRANSFO détermine l'ensemble des représentations du problème en appliquant toutes les transformations possibles à partir de la représentation initiale. Chaque représentation est caractérisée par les valeurs b_1, b_2, b_3 et v. Ce module construit une arborescence des représentations. La racine de l'arbre est la représentation initiale. Chaque nœud est associé à une représentation R donnée. Il a au plus 4 fils correspondant aux représentations images de R par les transformations possibles : $T_c(O\vec{i}), T_d(O\vec{i}), T_c(O\vec{j}), T_d(O\vec{j})$. Pour les problèmes définis

dans $\mathbf{R}^2$, un nœud a au plus 2 fils, associés aux transformations T_c et T_d liées au seul axe possible : $(O\vec{i})$.

— le module RESEAU parcourt l'arbre des représentations et, pour chaque nœud de cet arbre associé à une représentation valide, détermine un ensemble de réseaux systoliques correspondants. Pour cela, l'utilisateur indique une direction d'allocation choisie et les procédures du module calculent le nombre de cellules, les dessinent sur l'écran ainsi que les flots entre ces cellules et les données sur ces flots. Il réalise également la simulation d'exécution en déplaçant les données sur les flots à la demande de l'utilisateur.

— le module GRAPHIC définit l'ensemble des procédures de traitement graphique utilisées pour la visualisation des réseaux sur l'écran. Il utilise les opérations de base de la librairie graphique du bitmap ainsi que les opérations de manipulation des fontes de caractères pour l'impression des noms et des indices des données sur l'écran.

— les modules TRAITEXPR et TRAITCOND contiennent les procédures de manipulation des expressions arithmétiques et des conditions définissant les bords du réseau.

— le module SYSMAIN contient le programme principal.

Nous prévoyons de faire une présentation plus détaillée du logiciel et de son utilisation dans un manuel d'utilisation rédigé ultérieurement.

III. Limites actuelles

Les limites actuelles du logiciel se situent à deux niveaux :

— certains points décrits dans la méthode ne sont pas réalisés.

En particulier, il n'est pas possible de résoudre des problèmes avec des contraintes, soit d'accès aux données, soit de communications (résultat élémentaire utilisé comme donnée pour le calcul d'un autre résultat). Cela signifie que toutes les représentations obtenues par application de transformations sont valides.

Le logiciel ne traite que les problèmes dont le domaine est défini par des bornes constantes. De plus, les réseaux qu'il sait déterminer sont

caractérisés par un seul type de cellules. Cela signifie que les résultats à calculer sont les résultats de la récurrence, c'est-à-dire que le sous-domaine D_2 est vide (*cf.* chapitre 4 II.1).

— l'utilisation d'un écran graphique impose des restrictions.

Le réseau calculé n'est pas visualisé si le nombre de cellules est trop grand pour permettre son affichage en totalité sur l'écran.

Lorsque la distance entre deux cellules est trop petite pour imprimer une donnée sur le flot correspondant, les données de la suite ne sont pas affichées.

Dans les deux cas, un message apparait sur l'écran pour indiquer à l'utilisateur que les limites de l'écran ne permettent pas la visualisation, soit du réseau, soit des données d'une suite.

IV. Extensions prévues

La version de base de SYSTOL existante actuellement peut être améliorée et enrichie pour permettre une utilisation plus agréable et une extension à d'autres types de réseaux.

Pour faciliter son utilisation sont prévus :

— une amélioration de l'interface avec l'utilisateur (menus, commentaires plus importants, etc),

— des commandes permettant la mémorisation sur fichier de la spécification systolique d'un problème, de l'arborescence des représentations ou des réseaux obtenus. Ceci évitera à l'utilisateur d'entrer plusieurs fois les données et au logiciel de recalculer des résultats (arbre ou réseaux).

— pour chaque réseau, le dessin d'une cellule type avec indication de la fonction de calcul réalisée par une cellule à partir des données disponibles sur les flots d'entrée.

— des outils permettant d'obtenir d'autres formes d'un même réseau, par application de transvections (*cf.* chapitre 4 V.2).

— des outils permettant de grossir un réseau ou une partie d'un réseau afin de pouvoir visualiser des réseaux non dessinés actuellement en raison des limitations dues à la taille de l'écran.

— l'évaluation des performances d'un réseau (temps d'amorçage, temps total d'exécution, nombre de cellules actives, etc). Cela permettra à l'utilisateur de choisir les réseaux qui lui paraissent les plus performants, en fonction de ses propres critères.

— l'intégration, dans le logiciel, de la commande de copie d'écran.

Pour étendre son domaine d'utilisation sont prévus :

— de permettre que les bornes du domaine ne soient plus constantes mais définies en fonction des autres bornes.

— de traiter des problèmes dont le domaine est réunion de deux sous-domaines. De tels problèmes conduisent à des réseaux constitués de deux types de cellules.

De plus, les extensions de la méthode à d'autres classes de problèmes (avec des énoncés différents), à d'autres types de transformations (quasi-affines) ou à d'autres types de fonctions d'allocation (quasi-linéaires), donneront également lieu à des extensions au niveau du logiciel.

V. Un exemple d'utilisation du logiciel

Le logiciel est appelé au moyen de la commande SYSRUN. Pour l'utilisateur, son exécution se décompose en deux étapes.

La première étape est la saisie, à partir du clavier, de la spécification systolique du problème, réalisée par un dialogue interactif. Le processus est décrit pour le produit de convolution à la figure 5.2.

La deuxième étape est le parcours de l'arbre des représentations et l'affichage des réseaux. Le logiciel indique sur quelle représentation il travaille et demande, par un dialogue interactif avec l'utilisateur, de préciser successivement les directions d'allocation choisies pour cette représentation. Pour chaque direction, il affiche le réseau obtenu sur l'écran bitmap et demande à l'utilisateur s'il souhaite l'imprimer ou en effectuer une simulation de fonctionnement.

Il indique le cas échéant, que la direction choisie est impossible (abscisse temporelle nulle), qu'il ne peut pas dessiner le réseau (trop de cellules) ou

un flot (coupure de cellules) ou qu'il ne peut pas afficher certaines données (espacement entre cellules trop petit).

Cette deuxième étape est illustrée figure 5.3. Les réseaux obtenus pour cette session sont représentés aux figures 5.4 à 5.8 en utilisant un programme de "hard copy".

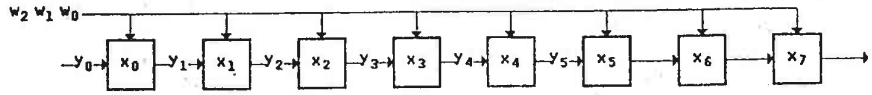

```
$ sysrun

 ATTENTION: decrire les expressions directement sous forme postfixee


*** Definition de la dimension du probleme
    Valeur (2 ou 3) : 2

*** Definition des bornes sur la composante i
    Borne inferieure (terminee par $) : 0$
    Borne superieure (terminee par $) : 5$

*** Definition des bornes sur la composante k
    Borne inferieure (terminee par $) : 0$
    Borne superieure (terminee par $) : 2$

*** Definition du nombre de ressources
    Valeur (1..20) : 3

*** Definition des familles de ressources (1 lettre minus.)

    Nom du resultat : y

    Nom et dimension des  2 familles de donnees
    Dim =1 donnee simple, =2 suite simple, =3 suite double

     1 Nom       : w
       Dimension : 2

     2 Nom       : x
       Dimension : 2

    Definir la fonction d'utilisation du resultat y
    Donner la fonction i$

    Definir la fonction d'utilisation de la ressource w
    Donner la fonction k$

    Definir la fonction d'utilisation de la ressource x
    Donner la fonction ik+$
```

FIGURE 5.2. La saisie de la spécification

```
*** REPRESENTATION : Initiale

  Direction d'allocation choisie (unimodulaire) : 1 -1

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : n

  Autre direction d'allocation (o/n) : o

  Direction d'allocation choisie (unimodulaire) : 0 1

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : n

  Autre direction d'allocation (o/n) : n

*** REPRESENTATION : TcOi

  Direction d'allocation choisie (unimodulaire) : 1 1

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : n

  Autre direction d'allocation (o/n) : n

*** REPRESENTATION : TcOi TcOi

  Direction d'allocation choisie (unimodulaire) : 1 0

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : n

  Autre direction d'allocation (o/n) : o

  Direction d'allocation choisie (unimodulaire) : 0 1

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : n

  Autre direction d'allocation (o/n) : n

*** REPRESENTATION : TcOi Tec

  Direction d'allocation choisie (unimodulaire) : 1 0

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : n

  Autre direction d'allocation (o/n) : n

*** REPRESENTATION : TdOi

  Direction d'allocation choisie (unimodulaire) : 1 0

  Simulation du reseau. A chaque demande s'affiche l'etat du reseau en t,t+1..

  Voulez-vous visualiser la simulation (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : o
  Voulez-vous continuer (o/n) : n

  Autre direction d'allocation (o/n) : n

   **** Voulez-vous revoir les differents reseaux (o/n)? n
```

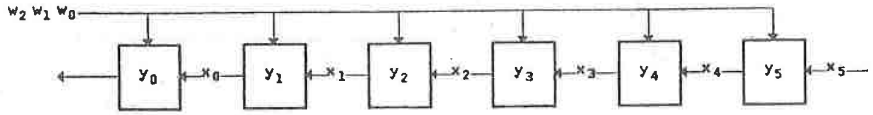
FIGURE 5.3. Le traitement des représentations et le dessin des réseaux

SYSTOL



(a) pour $\vec{\xi} = (1, -1)_{B.C.}$

SYSTOL



(b) pour $\vec{\xi} = (0, 1)_{B.C.}$

FIGURE 5.4. Réseaux pour la représentation R_0

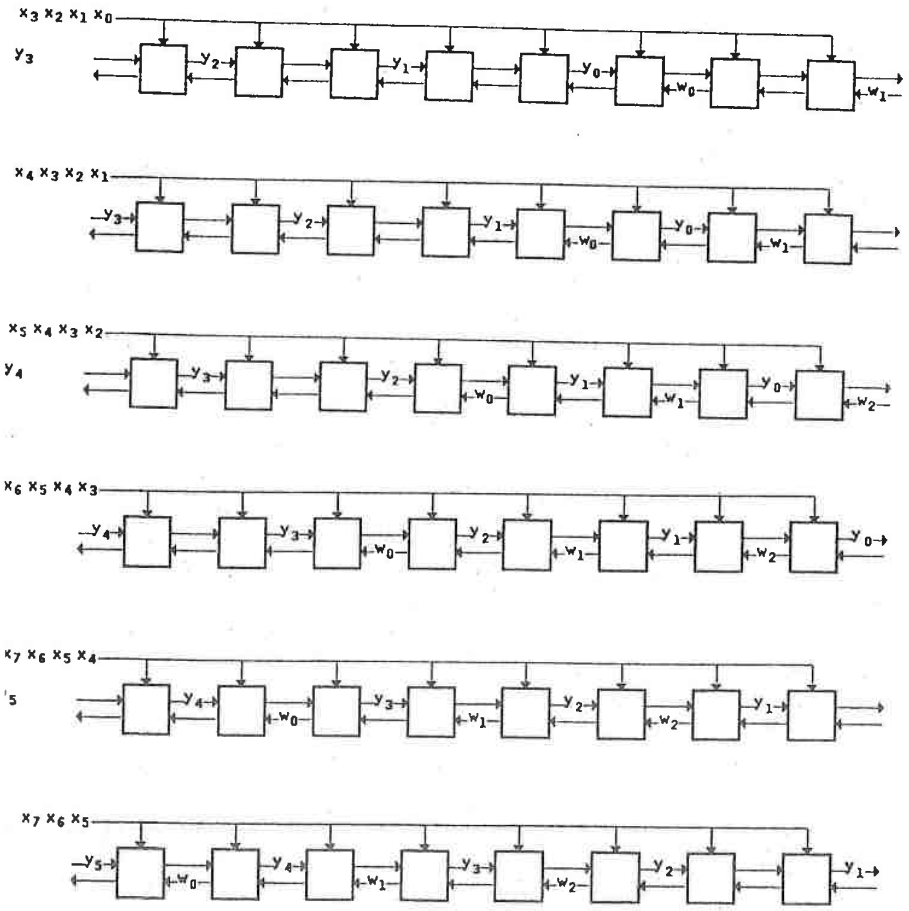
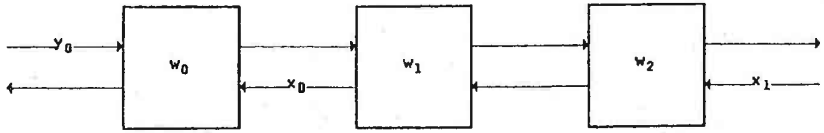
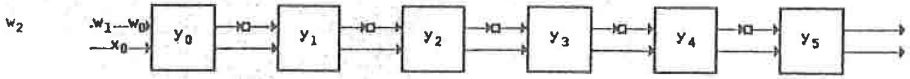


FIGURE 5.5. Réseau pour la représentation $T_c(O\vec{i})_{R_0}(R_0)$ avec $\vec{\xi} = (1,1)_{B.C.}$
Simulation de fonctionnement



(a) pour $\vec{\xi} = (0,1)_{B.C.}$



(b) pour $\vec{\xi} = (1,0)_{B.C.}$

FIGURE 5.6. Réseaux pour la représentation $(T_c(O\vec{i}) \circ T_c(O\vec{i}))_{R_0}(R_0)$

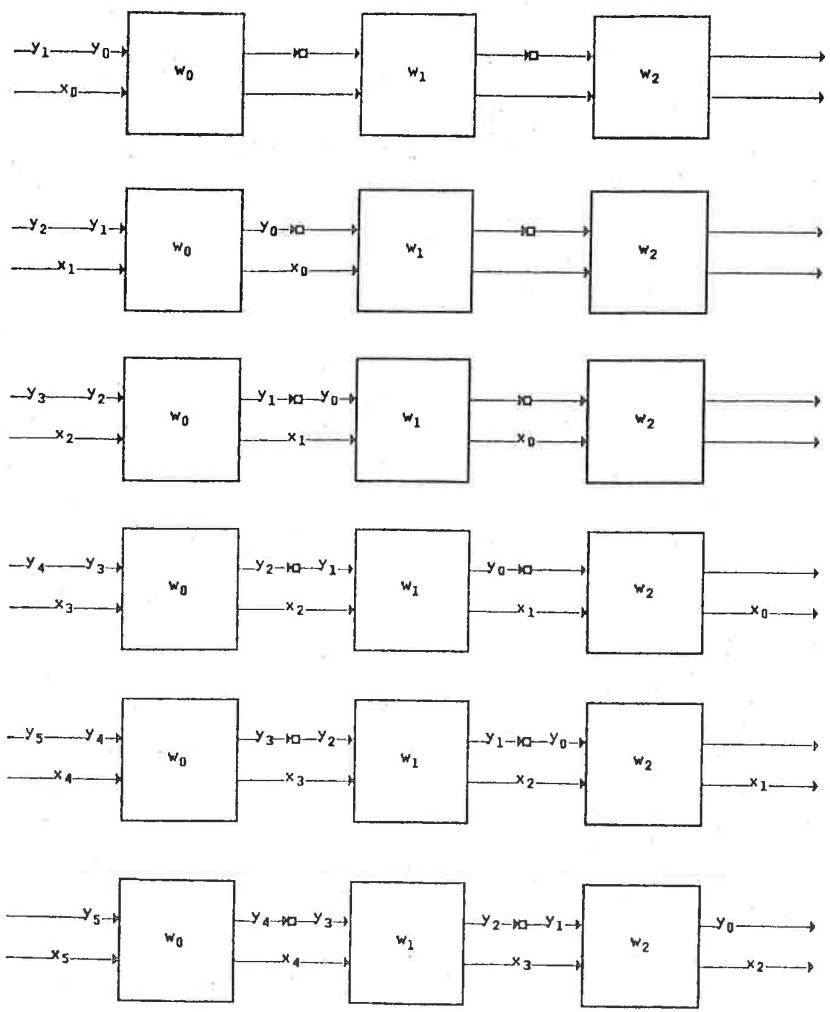


FIGURE 5.7. Réseau pour la représentation $(T_c(O\vec{i}) \circ T_{ec})_{R_0}(R_0)$
avec $\vec{\xi} = (1,0)_{B.C.}$
Simulation de fonctionnement

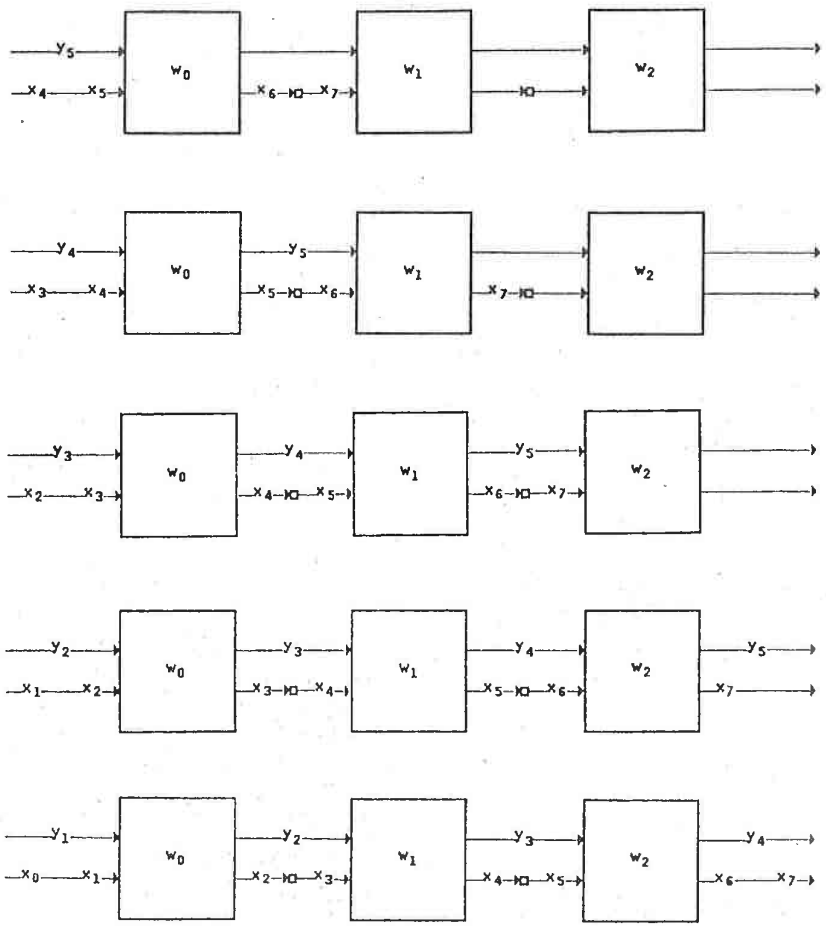


FIGURE 5.8. Réseau pour la représentation $T_d(O\vec{i})_{R_0}(R_0)$
avec $\vec{\xi} = (1,0)_{B.C.}$
Simulation de fonctionnement

CHAPITRE 6

AUTRES APPROCHES DE LA CONCEPTION D'ALGORITHMES SYSTOLIQUES

Les premiers travaux de KUNG sur les algorithmes parallèles pour circuits VLSI ont suscité un très grand intérêt. De nombreux algorithmes systoliques ont été proposés pour la résolution de problèmes tels que le produit de convolution [KUN 82], le produit matriciel, la décomposition de matrice sous forme LU, la résolution de système triangulaire [KUL 80], le tri de tableau, l'évaluation de récurrences [KUN 79], les problèmes de connexité [TCM 83], la détection de mots dans la parole continue [AFQ 83], la recherche de sous-chaines [FOK 80], etc.

Outre ces travaux portant sur des propositions d'algorithmes systoliques adaptés à la résolution d'un problème particulier, certains auteurs comme D. MOLDOVAN, W. MIRANKER et A. WINKLER, P. QUINTON se sont intéressés eux aussi à la conception d'algorithmes systoliques.

Etudions les similitudes et les différences entre ces différentes approches et celle proposée dans cette thèse, en nous appuyant sur les critères décrits au paragraphe I.

On peut déjà noter que ces trois approches ne s'intéressent qu'à l'obtention de réseaux systoliques purs contrairement à notre méthode qui permet de déterminer également les réseaux semi systoliques avec diffusion de données.

I. Critères de comparaison

Avant de décrire chacune des trois méthodes étudiées, nous définissons quelques critères qui vont guider leur comparaison. Nous pourrons ainsi,

après une présentation des différentes méthodes, les comparer (au paragraphe V), en fonction des critères définis et mieux cerner l'apport de la méthode proposée dans cete thèse.

Nous distingons deux types de critères :

— les critères liés directement à la méthode,
  . quel est son point de départ ?
  . quels sont les concepts qu'elle utilise ?
  . est-elle constructive ?

— les critères liés à l'utilisation de la méthode,
  . est-il possible de réaliser un logiciel d'aide à la conception de réseaux systoliques s'appuyant sur cette méthode dans sa description actuelle ?
  . quelle est la diversité des réseaux obtenus par le logiciel, s'il peut être construit ?

II. Approche de Moldovan [MOL 83]

La méthode de conception d'algorithmes systoliques proposée par MOLDOVAN est basée sur une transformation des algorithmes séquentiels, définis dans un langage de haut niveau (Fortran ou Algol) et exprimant des boucles, en un réseau VLSI correspondant.

Les performances d'un circuit VLSI pouvant être amoindries par des communications trop complexes, une utilisation efficace est obtenue avec uniquement des connexions locales. L'auteur se limite donc aux réseaux systoliques purs et la première étape de la méthode consiste à supprimer les diffusions de données apparaissant dans l'algorithme en pipelinant les données diffusées. Ainsi pour le produit de matrices carrées de dimension n dont un algorithme séquentiel en Fortran est :

DO 10 $k = 1, n$
DO 10 $i = 1, n$
DO 10 $j = 1, n$ (1)
$c(i,j) = c(i,j) + a(i,k) * b(k,j)$
10 **CONTINUE**

la diffusion est évidente en raison de l'absence d'un des trois indices de boucle sur les variables a, b et c. Pour pipeliner les données, on complète les indices correspondants et on introduit ainsi des variables artificielles afin que toute donnée n'ait qu'une utilisation. Le nouvel algorithme est

DO 10 $k = 1, n$
DO 10 $i = 1, n$
DO 10 $j = 1, n$
$a^{j+1}(i,k) = a^j(i,k)$
$b^{i+1}(k,j) = b^i(k,j)$
$c^{k+1}(i,j) = c^k(i,j) + a^j(i,k) * b^i(b,j)$
10 **CONTINUE**

L'algorithme est caractérisé par l'ensemble $\mathcal{L}^n$ des indices de n boucles imbriquées. Ici, on a $\mathcal{L}^3 = \{(k,i,j) \mid 1 \leq k \leq n, 1 \leq i \leq n, 1 \leq j \leq n\}$. Il correspond au domaine associé au problème tel que nous le définissons au chapitre 4.

La deuxième étape consiste à déterminer l'ensemble des vecteurs de dépendance de l'algorithme. Si un pas de l'itération, caractérisé par un n-uplet d'indices $\overline{I} = (i_1, \ldots, i_n) \in \mathcal{L}^n$, utilise une donnée calculée par un pas de l'itération, caractérisé par $\overline{J} = (j_1, \ldots, j_n) \in \mathcal{L}^n$, avec $\overline{I} < \overline{J}$ où $<$ est la relation d'ordre lexicographique sur l'ensemble $\mathcal{L}^n$, alors on définit un vecteur de dépendance associé à cette donnée par $\overline{d} = \overline{J}^t - \overline{I}^t$. Ces vecteurs de dépendance peuvent être constants ou fonctions des éléments de $\mathcal{L}^n$.

Ainsi, pour l'algorithme ci-dessus, la donnée $c^k(i,j)$ calculée au pas défini par le triplet $(k-1,i,j)$ est utilisée au pas (k,i,j). Ceci définit un premier vecteur de dépendance $d_1 = (k,i,j) - (k-1,i,j) = (1,0,0)$. De même, le pas (k,i,j) utilise la donnée $a^j(i,k)$ calculée au pas $(k,i,j-1)$, ainsi que la donnée $b^i(k,j)$ calculée au pas $(k,i-1,j)$. Les deux autres vecteurs de dépendance du problème sont donc $d_2 = (0,0,1)$ et $d_3 = (0,1,0)$.

L'étape suivante consiste à appliquer, sur la structure $< \mathcal{L}^n, R >$ où R est l'ordre imposé par les vecteurs de dépendance sur l'ensemble $\mathcal{L}^n$, une transformation T définie par $T :< \mathcal{L}^n, R > \rightarrow < \mathcal{L}^n_T, R_T >$ monotone et bijective. La fonction T est partitionnée en deux fonctions Π et S définies par

$$\Pi : \mathcal{L}^n \rightarrow \mathcal{L}^k_T \qquad k < n$$
$$S : \mathcal{L}^n \rightarrow \mathcal{L}^{n-k}_T$$

L'entier k, qui dimensionne Π et S, est tel que la fonction Π établisse à elle seule l'ordre R_T. Ainsi, les k premières coordonnées des éléments $\overline{J} \in \mathcal{L}_T^n$ sont liées au temps tandis que les $n-k$ dernières sont liées aux propriétés géométriques de l'algorithme. Notons que dans la pratique, il faut $n-k \leq 2$ pour avoir des réseaux planaires.

Dans le cas où l'algorithme, constitué de n boucles, est caractérisé par m vecteurs de dépendance constants, $D = [\overline{d}_1, \overline{d}_2, \ldots, \overline{d}_m]$, la transformation T est choisie linéaire, c'est-à-dire $\overline{J} = T.\overline{I}$ Si l'on note $\overline{\delta}_j$ le vecteur de dépendance $\overline{d}_j$ après transformation, $\overline{\delta}_j = T\overline{d}_j$, le système à résoudre est $TD = \Delta$ où $\Delta = [\overline{\delta}_1, \overline{\delta}_2, \ldots, \overline{\delta}_m]$

Des conditions nécessaires et suffisantes pour qu'il existe une transformation T valide pour un tel algorithme sont

(i) $\overline{\delta}_j \equiv \overline{d}_j$ modulo c_j, c_j étant le PGCD des éléments de $\overline{d}_j$
(ii) le système $TD = \Delta$ a une solution
(iii) le premier élément non nul de $\overline{\delta}_j$ est positif.

Ainsi, pour le problème du produit matriciel défini ci-dessus, les vecteurs de dépendance sont définis par

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

donc une transformation linéaire T est telle que $T = \Delta$. Le premier élément non nul de $\overline{\delta}_j$ étant positif, nous prenons $\Pi\overline{d}_i > 0$ pour offrir le maximum de simultaneité. Notons

$$T = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ \cdots & \cdots & \cdots \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix}$$

on peut prendre $k = 1$ pour dimensionner Π et S

Donc $\Pi\overline{d}_i = t_{1i} > 0$. On prend donc pour $t_{1i}, i = 1 \ldots 3$, les plus petites valeurs positives, c'est-à-dire $t_{11} = t_{12} = t_{13} = 1$.

La fonction S est déterminée en tenant compte du fait que T est une bijection dont la matrice est constituée d'entiers, c'est-à-dire $det(T) = \pm 1$.

Une solution parmi les nombreuses possibles est

$$T = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

De cette transformation de l'ensemble des indices, on déduit alors le réseau systolique associé de la manière suivante :

— les fonctions calculées par les cellules sont déduites des expressions mathématiques de l'algorithme. Un algorithme de la forme (1) contient des instructions exécutées à chaque point de l'ensemble $\mathcal{L}^n$. Les cellules sont donc toutes identiques, excepté bien sûr les cellules périphériques effectuant les entrées-sorties. Si les calculs de la boucle sont trop importants, la boucle peut être décomposée en plusieurs boucles plus simples. Le réseau correspondant nécessite alors plusieurs cellules différentes.

— la géométrie du réseau est déduite de la fonction $S : \mathcal{L}^n \rightarrow \mathcal{L}_T^{n-k}$. Le numéro d'identification de chaque cellule est donné par $S(\overline{I}) = (J^{k+1}, \ldots, J^n)$ pour $I \in \mathcal{L}^n$. Les interconnexions entre les cellules sont déduites des $n-k$ dernières composantes des vecteurs de dépendance $\overline{\delta}_j$ obtenus après transformation : $\overline{\delta}_j^S = S(\overline{I}+\overline{d}_j) - S(\overline{I})$. Lorsque T est linéaire $\overline{\delta}_j^S = S\overline{d}_j$. Pour chaque cellule, les vecteurs $\overline{\delta}_j^S$ indiquent le numéro d'identification de la cellule destination pour la variable associée à ce vecteur.

— le déroulement du réseau dans le temps est donné par $\Pi = \mathcal{L}^n \rightarrow \mathcal{L}_T^k$ Le calcul élémentaire correspondant à $\overline{I} \in \mathcal{L}^n$ est exécuté à l'instant $\Pi(\overline{I})$. Le temps de communication pour un flot de données associé au vecteur de dépendance $\overline{d}_j$ est donné par $\Pi(\overline{I}+\overline{d}_j) - \Pi(\overline{I})$ qui se réduit à $\Pi(\overline{d}_j)$ quand la transformation T est linéaire.

En prenant pour l'entier k dimensionnant Π et S la plus petite valeur possible, on augmente le nombre d'opérations parallèles aux dépens du nombre de cellules.

REMARQUE. — Aucune précision n'est donnée sur la signification temporelle de la fonction Π dans le cas où $k > 1$.

Ainsi, par le produit matriciel défini en (1) avec comme transformation linéaire

$$T = \begin{pmatrix} 1 & 1 & 1 \\ \cdots & \cdots & \cdots \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

la fonction S est définie par

$$S\begin{pmatrix} k \\ i \\ j \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} k \\ i \\ j \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix}$$

Le réseau est donc un réseau bidimensionnel carré représenté à la figure 6.1 pour $n = 4$.

Les circulations de données sont définies par $S\bar{d}_j$. Pour les données c_{ij}, le vecteur de dépendance est $\bar{d}_1 = (1,0,0)^t$, d'où $S\bar{d}_1 = \binom{0}{0}$. Les données c_{ij} sont donc constantes des cellules. Le vecteur de dépendance des données a_{ik} étant $\bar{d}_2 = (0,0,1)^t$, on a $S\bar{d}_2 = \binom{0}{1}$. Les données a_{ik} circulent donc horizontalement de gauche à droite. Pour les données b_{kj}, on a $S\bar{d}_3 = \binom{1}{0}$. Les données b_{kj} circulent donc verticalement de haut en bas.

L'article n'expose pas explicitement comment sont organisées les données sur les flots (indiçage, espacement des données, etc) en utilisant la fonction Π.

Une comparaison de l'approche de Moldovan et de celle proposée dans cette thèse, amène les remarques suivantes :

— le point de départ de la méthode de Moldovan est d'exprimer le problème par un algorithme séquentiel alors que nous proposons comme point de départ une spécification formelle du problème par un système d'équations récurrentes.

— la notion de vecteur de dépendance de Moldovan est similaire à notre notion de vecteur générateur. Elle permet de définir les différents calculs utilisant une même donnée.

La détermination de ces vecteurs nécessite une transformation préalable de l'algorithme qui supprime la diffusion des données alors que les vecteurs générateurs sont déterminés directement à partir de la spécification systolique du problème.

De plus, il ne propose pas de démarche sytématique permettant d'effectuer cette transformation automatiquement.

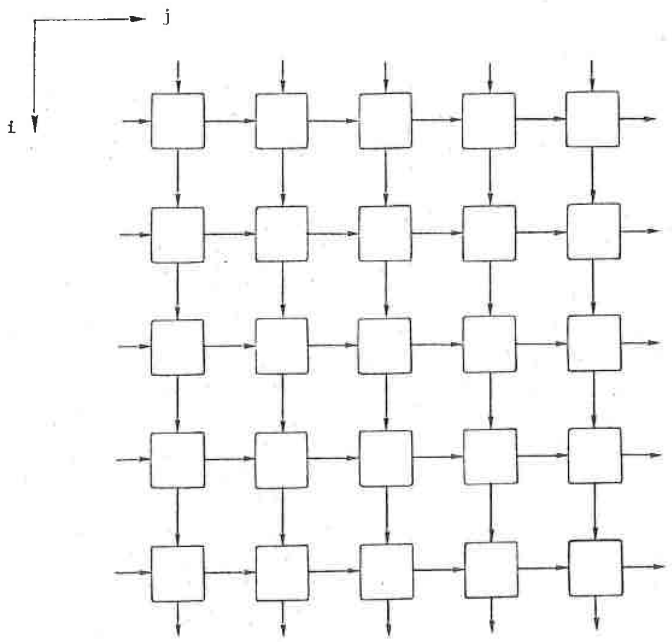


FIGURE 6.1. Réseau pour le produit matriciel avec $T = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

— la fonction S est la fonction d'allocation de notre méthode. Elle définit la référence de la cellule sur laquelle s'exécute chacun des calculs élémentaires.

— la fonction Π définit les calculs simultanés. Elle correspond à l'équation, dans la base canonique, des hyperplans d'équation $t = c$, $c \in \mathbf{N}$, dans l'espace-temps associé à une représentation.

— sa méthode n'indique pas comment obtenir de manière automatique un ensemble de transformations $T = \begin{bmatrix} \Pi \\ S \end{bmatrix}$ intéressantes. Elle ne permet donc pas dans l'état actuel de concevoir un système d'aide à la conception, contrairement à notre approche.

III. Approche de Miranker et Winkler [MIW 82]

La méthode proposée par MIRANKER et WINKLER est basée sur l'expression du problème par un graphe caractérisant les dépendances entre les données, sur lequel on applique une fonction permettant de définir l'instant et le lieu d'exécution de chaque calcul.

Comme Moldovan, le point de départ de leur méthode est de définir le problème par un algorithme séquentiel exprimé en langage de haut niveau (ici Fortran) et de le transformer en une forme canonique qui supprime la diffusion de données. Pour cela, ils souhaitent eux-aussi indexer chaque variable avec tous les indices de boucles où elle est imbriquée.

Ainsi à un algorithme du produit matriciel $C = A \times B$ définit par

DO 10 $i = 1, n$
DO 10 $j = 1, n$
DO 10 $k = 1, n$
$c(i,j) = c(i,j) + a(i,k) \times b(k,j)$
10 **CONTINUE**

est associé la forme canonique suivante :

DO 10 $i, j, k = 1, n$
$a(i,j,k) = a(i,j-1,k)$ (2)
$b(i,k,j) = b(i-1,k,j)$
$c(i,j,k) = c(i,j,k-1) + a(i,j,k) \times b(i,k,j)$
10 **CONTINUE**

REMARQUE. — Cette forme canonique est identique à celle obtenue par Moldovan, seul l'indiçage est différent.

Comme Moldovan, ils se limitent aux algorithmes constitués de n boucles imbriquées et déterminent, à partir de la forme canonique du problème,

l'ensemble $\Gamma \subset \mathbf{Z}^n$ des n-uplets des valeurs des indices de boucles. L'ensemble Γ correspond à l'ensemble $\mathcal{L}^n$ défini par Moldovan et au domaine D défini dans notre méthode.

Γ est considéré comme un ensemble de nœuds. Un nœud x de Γ correspond à un pas de l'itération. Il utilise des résultats calculés par d'autres nœuds. Ils appellent domaine de dépendance D_x de x, l'ensemble des nœuds y calculant un résultat utilisé par le nœud x et définissent pour le nœud x l'ensemble de ses vecteurs de dépendance : $\overrightarrow{xy}$ pour $y \in D_x$. Γ^* est l'ensemble de vecteurs de dépendance associés à tous les nœuds de Γ. Si l'on considère Γ^* comme un ensemble d'arcs orientés sur Γ, (Γ, Γ^*) définit un graphe orienté.

REMARQUE. — La notion de vecteur de dépendance est similaire à celle de Moldovan excepté que Moldovan inverse origine et destination du vecteur.

Lorsque les vecteurs de dépendance sont les mêmes pour tous les nœuds de Γ, on dit que Γ^* est une structure systolique. Son implantation physique est un réseau systolique.

Deux structures systoliques dont les vecteurs de dépendance sont respectivement $d_1, \ldots, d_k$ et $\Delta_1, \ldots, \Delta_k$ sont dites isomorphes s'il existe $A \in \mathrm{Gl}(n, \mathbf{Z})$ telle que $A \times d_i = \Delta_i, 1 \leq i \leq k$ ($\mathrm{Gl}(n, \mathbf{Z})$ est le groupe des matrices inversibles à coefficients entiers c'est-à-dire de déterminant à ± 1).

Le problème est maintenant de déterminer une réalisation physique d'une structure, définir pour chaque noeud Γ où et à quel instant le calcul correspondant s'exécute. Cela est réalisé par une fonction de la forme

$$\Gamma \subset \mathbf{Z}^n \longrightarrow^{(t,x,y,z)} \mathbf{Z}^4$$

où t représente l'instant où le noeud est calculé et (x, y, z) les coordonnées de l'endroit où il est calculé.

Les dépendances de données imposent des conditions sur la fonction. Ainsi lorsqu'il y a plusieurs processeurs, ce qui est le cas pour les réseaux systoliques, la condition suivante doit être réalisée :

Supposons le noeud $I \in \Gamma$ calculé par le processeur (x, y, z) au temps t et le noeud $I' \in \Gamma$ calculé par le processeur (x', y', z') au temps t'.

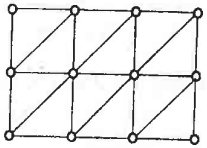
Si I' référence I alors il faut du temps pour que l'information transite de (t, x, y, z) à (t', x', y', z').

Ceci est exprimé par l'inégalité suivante appelée "time-like" condition pour le vecteur de dépendance $(\Delta t, \Delta x, \Delta y, \Delta z)^T$:

$$c\Delta t^2 - \Delta x^2 - \Delta y^2 - \Delta z^2 \geq 0$$

où c est la vitesse de la lumière, $\Delta t = t' - t, \Delta x = x' - x, \Delta y = y' - y, \Delta z = z' - z$.

La réalisation d'une structure systolique est un tableau régulier de processeurs organisé de façon planaire. Les connexions correspondent aux vecteurs de dépendances Δ_i. Ils définissent la forme la plus générale du tableau systolique à deux dimensions, appelée tableau systolique planaire universel, comme étant



Tout réseau systolique associé à un problème est inclus dans ce tableau systolique planaire universel. Etant donné une structure caractérisée par ses vecteurs de dépendance $v_1 \ldots v_k$, sa réalisation consiste en une fonction de la forme $(v_1 \ldots v_k) \rightarrow \mathbf{Z}^3$ où $\mathbf{Z}^3$ = temps $\times$ (x, y), (x, y) étant les composantes spatiales.

Pour obtenir un réseau inclus dans le tableau systolique universel, ces composantes spatiales doivent être de la forme $\pm(0,1)^T, \pm(1,0)^T, \pm(1,1)^T$.

Pour la forme canonique du produit matriciel donné en (2), les vecteurs de dépendance sont

$$d_1 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \qquad d_2 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \qquad d_3 = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

Une structure isomorphe est donnée par $A \in \mathrm{Gl}(3, \mathbf{Z})$ telle que

$$A(d_1, d_2, d_3) = (\Delta_1, \Delta_2, \Delta_3)$$

Par exemple

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 \\ \cdots & \cdots & \cdots \\ 0 & -1 & -1 \\ 0 & 0 & -1 \end{pmatrix}.$$

Les vecteurs Δ_i obtenus vérifient bien la "time-like" condition. Leur première composante correspond au temps, les deux dernières correspondent aux coordonnées planaires. Ces vecteurs $\binom{0}{0}, \binom{-1}{0}, \binom{-1}{-1}$ traduisent respectivement le fait qu'une donnée est constante de cellule, une transite de droite à gauche et une diagonalement de droite à gauche comme l'indique la figure 5.2.

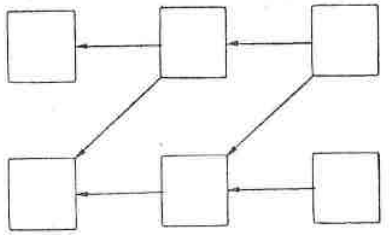


FIGURE 5.2

On remarque de nombreuses similitudes entre cette approche et celle proposée par Moldovan :

- le point de départ est également un algorithme séquentiel qu'il faut transformer pour obtenir une forme canonique supprimant la diffusion de données.
- les vecteurs de dépendance, qui sont déduits de cette forme canonique, sont aussi les vecteurs générateurs décrits aux chapitre 3 et 4 de notre méthode.
- l'ensemble des nœuds Γ associé au problème correspond également au domaine défini dans notre méthode.

Dans son état actuel, la méthode proposée n'indique pas comment déterminer un ensemble de fonctions $A \in \mathrm{Gl}(n, \mathbf{Z})$ de manière systématique, afin d'obtenir une famille de réseaux systoliques correspondants. Ceci rend

difficile la réalisation d'un système d'aide à la conception d'algorithmes systoliques.

III. Approche de Quinton [QUI 83]

La méthode proposée par QUINTON se décompose en 3 étapes :

— exprimer le problème par un ensemble d'équations récurrentes uniformes sur un domaine de calcul $D \subset \mathbb{Z}^n$, lorsque cela est possible.

— définir à partir de cet ensemble d'équations récurrentes uniformes une fonction de temps permettant d'ordonner les calculs.

— projeter le domaine de calcul sur une machine finie en utilisant une fonction d'allocation linéaire.

Un système d'équations récurrentes uniformes est un système de la forme :

$$(3)\begin{cases} u_1(z) = f(u_1(z+\theta_1), u_2(z+\theta_2), \ldots, u_p(z+\theta_p) \\ u_2(z) = u_2(z+\theta_2) \\ \quad\vdots \\ u_p(z) = u_p(z+\theta_p) \end{cases}$$

où les vecteurs $\theta_i \in \Theta$, appelés *vecteurs de dépendance* sont indépendants de z. Ils définissent, pour un point du domaine, en quels points il doit prendre ses valeurs d'entrée.

L'ensemble $D \subset \mathbb{Z}^n$ des points z pour lesquels le système ci-dessus est défini, est appelé le *domaine* de calcul. La paire (D, Θ) est appelée le *graphe de dépendance.*

Ainsi, le calcul d'un produit de convolution défini par

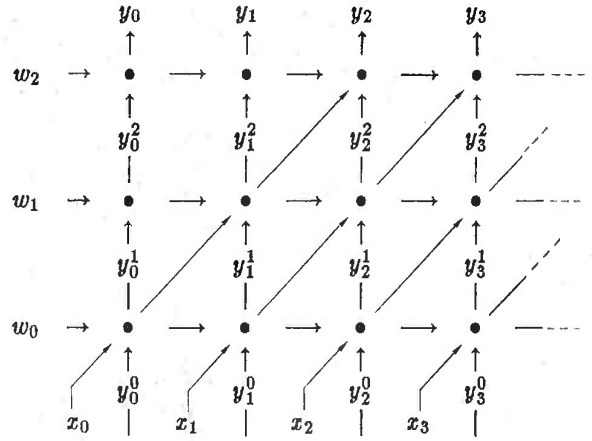
$$y_i = \sum_{k=0}^{K} w_k \times x_{i-k}, \quad i \geq 0$$

peut être défini par le système d'équations récurrentes uniformes suivant :

$$(4)\begin{cases} y(i,k) = y(i,k-1) + w(i-1,k) \times x(i-1,k-1) \\ w(i,k) = w(i-1,k) \\ x(i,k) = x(i-1,k-1) \end{cases}$$

Les vecteurs de dépendance sont $\theta_y = (0,-1), \theta_w = (-1,0), \theta_x = (-1,-1)$ et le domaine de calcul est $D = \{(i,k) \mid i \geq 0, \quad 0 \leq k \leq K\}$.

Le graphe de dépendance (D, Θ) peut être représenté, pour $k = 2$, par



Les points (•) représentent les éléments du domaine de calcul D. Les flèches entre ces points représentent les vecteurs de dépendance. Ainsi les vecteurs diagonaux correspondent à θ_x : le calcul de $y(i,k)$ utilise la donnée x_{i-k} qui provient du calcul de $y(i-1,k-1)$ qui l'utilise également.

Si un autre choix pour la provenance des données est pris, on obtient d'autres vecteurs de dépendance. Un problème peut donc être décrit par plusieurs systèmes d'équations récurrentes uniformes différents.

Ainsi le produit de convolution décrit ci-dessus peut être défini par le système d'équations récurrentes uniformes

$$(5)\begin{cases} y(i,k) = y(i,k-1) + w(i-2,k) \times x(i-1,k-1) \\ x(i,k) = x(i-1,k-1) \\ w(i,k) = w(i-2,k) \end{cases}$$

REMARQUE. — L'algorithme de convolution décrit ici est différent de celui proposé au chapitre 3.

La deuxième étape consiste à déterminer, pour un système d'équations récurrentes uniformes, toutes les fonctions de temps possibles. Une fonction de temps est une fonction de $D \subset \mathbf{Z}^n$ dans $\mathbf{N}$ qui indique à quel instant chaque calcul doit être effectué. Une telle fonction doit vérifier la condition suivante :
si $x \in D$ dépend de $y \in D$, c'est-à-dire s'il existe un vecteur de dépendance $\theta_i = \overrightarrow{xy}$ alors $t(x) > t(y)$.

Les seules fonctions de temps utilisées dans la méthode sont celles appelées quasi-affines de la forme

$$t(x) = \lfloor \lambda^T x - \alpha \rfloor \quad \lambda \in \mathbf{R}^n, \quad \alpha \in \mathbf{R}$$

où $\lfloor u \rfloor$ indique le plus grand entier inférieur ou égal à u.

Une telle fonction est notée (λ, α). Si $\lambda \in \mathbf{Z}^n$ et $\alpha \in \mathbf{Z}$, la fonction est dite affine. En pratique les seules fonctions quasi-affines considérées sont celles à coefficients rationnels.

Lorsque le domaine D du problème est convexe, l'analyse convexe permet de déterminer toutes les fonctions de temps quasi-affines possibles. Pour cela, il utilise les définitions suivantes :

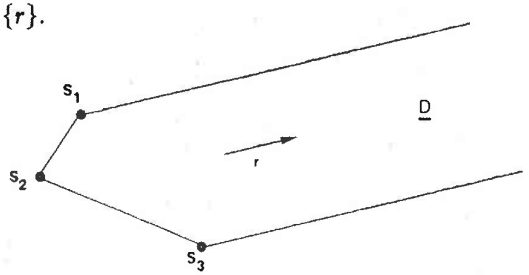
- le domaine D est le sous-ensemble des points de coordonnées entières d'un domaine polyhédral convexe de $\mathbf{R}^n$ appelé $\underline{D}$
- $\sum_{i=1}^n \mu_i \times x_i$ est une combinaison positive de points $x_1 \dots x_n$ de $\mathbf{R}^n$ si $\mu_i, i = 1, \dots, n$, est un réel non négatif
- $\sum_{i=1}^n \alpha_i \times x_i$ est une combinaison convexe de $x_1 \dots x_n$ si c'est une combinaison positive et si $\sum_{i=1}^n \alpha_i = 1$
- s est un sommet de $\underline{D}$ si s ne peut être exprimé comme une combinaison convexe de 2 points différents de $\underline{D}$
- r est un rayon de $\underline{D}$ si $\forall x \in \underline{D}, \forall \mu \in \mathbf{R}^+, \quad x + \mu r \in \underline{D}$
- un rayon de $\underline{D}$ est extrémal s'il ne peut être exprimé comme une combinaison positive d'autres rayons de $\underline{D}$
- l est une ligne de $\underline{D}$ si $\forall x \in \underline{D}, \forall \mu \in \mathbf{R}, x + \mu l \in \underline{D}$
- si $\underline{D}$ contient une ligne, $\underline{D}$ est appelé un cylindre.

On se limite aux domaines polyhédraux convexes qui ne sont pas des cylindres. Dans ce cas, l'ensemble des sommets S de $\underline{D}$ est unique ainsi que l'ensemble R des rayons extrémaux de $\underline{D}$, unique à un multiple scalaire non nul près.

On peut alors définir $\underline{D}$ comme le sous-ensemble des points x de $\mathbf{R}^n$ tel que $x = y + z$ où y est une combinaison convexe de sommets de S et z une combinaison positive de rayons de R.

La paire (S, R) est appelée un *système de générateurs* de $\underline{D}$. Elle peut être déterminée systématiquement à partir des inéquations définissant $\underline{D}$.

REMARQUE. — Le dessin ci-dessous représente un domaine polyhédral convexe $\underline{D}$ de $\mathbf{R}^2$ défini par 3 sommets $S = \{s_1, s_2, s_3\}$ et par un rayon $R = \{r\}$.



La paire (S, R) est un système de générateurs de $\underline{D}$ car tout point de $\underline{D}$ peut être défini de manière unique à partir de s_1, s_2, s_3 et r.

Le théorème constructif suivant permet alors de définir toutes les fonctions de temps quasi-affines possibles $t = (\lambda, \alpha)$:
$t = (\lambda, \alpha)$ est une fonction de temps quasi-affine pour (D, Θ) si

$$(i) \quad -\lambda^T \theta \geq 1 \quad \forall \theta \in \Theta$$

$$(ii) \quad \lambda^T r \geq 0 \quad \forall r \in R$$

$$(iii) \quad \alpha \leq \lambda^T s \quad \forall s \in S$$

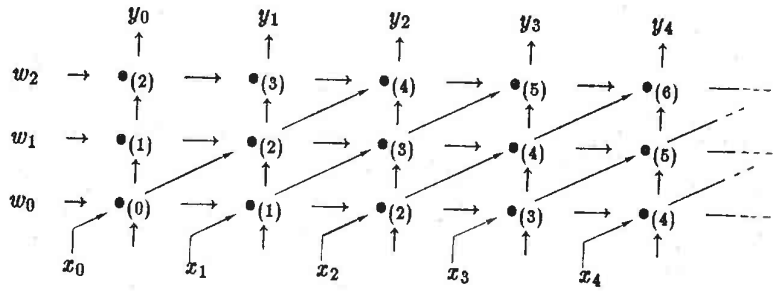
Ainsi pour le système d'équations récurrentes uniformes (4) définissant le produit de convolution, l'unique système de générateurs est

$$S = \{(0,0), (0,K)\} \quad \text{et} \quad R = \{(1,0)\}$$

L'application du théorème ci-dessus permet de caractériser les fonctions de temps (λ, α) par les conditions suivantes :

$\lambda^T = (\lambda_1, \lambda_2)$ vérifie $\lambda_1 \geq 1$, $\lambda_2 \geq 1$, $\lambda_1 + \lambda_2 \geq 1$
$\alpha \leq 0$

Une fonction de temps possible est donc définie par $\lambda^T = (1,1)$ et $\alpha = 0$, c'est-à-dire $t(i,k) = i+k$ et peut être représentée par le schéma suivant où l'instant d'exécution d'un calcul est indiqué à côté du temps correspondant.



La dernière étape de la méthode consiste à appliquer une fonction d'allocation pour projeter le domaine sur un ensemble fini, les cellules du réseau. Cette fonction a de D dans un sous-ensemble fini de $\mathbf{Z}^m$, où m est la dimension du réseau systolique obtenu, doit vérifier la condition suivante :

$$\forall x, y \in D \quad a(x) = a(y) \Rightarrow t(x) \neq t(y)$$

Cette condition garantit que deux calculs exécutés sur une même cellule ne sont pas simultanés.

La méthode se restreint à la détermination de fonctions d'allocations linéaires ou quasi-linéaires : a de $D \subset \mathbf{Z}^n$ dans $\mathbf{Z}^m$ est quasi-linéaire s'il existe une fonction a' de $\mathbf{Z}^n$ dans $\mathbf{Z}^m$ et un vecteur P de $\mathbf{Z}^m$ tels que $a(x) = a'(x) \bmod P$.

Lorsque t est une fonction de temps affine, l'idée est de projeter le graphe de dépendance parallèlement à une direction définie par un vecteur d'allocation non nul ξ de $\mathbf{Z}^n$ et non parallèle aux hyperplans de temps. Quand le domaine D est fini, le résultat de la projection $a(D)$ est évidemment fini. Le seul cas de domaine infini pris en compte ici concerne les domaines possèdant un seul rayon r.

A partir du choix d'une direction d'allocation non nulle $\xi \in \mathbf{Z}^n$, on détermine une fonction a de la manière suivante :

Supposons $\xi_n \neq 0$. Un point $(x_1 \ldots x_n)$ de $\mathbf{R}^n$ a pour décomposition $(y_1 \ldots y_n)$ sur la base $(e_1, \ldots, e_{n-1}, \xi)$ avec

$$\begin{cases} y_i = x_i - x_n(\xi_i/\xi_n), & \text{pour } 1 \leq i \leq n-1, \\ y_n = x_n/\xi_n \end{cases}$$

La fonction a est définie par $a(x_1 \ldots x_n) = \xi_n(y_1 \ldots y_{n-1})$ et le théorème suivant indique comment construire la fonction d'allocation à partir de a :
Supposons $\lambda^T r \neq 0$
(i) si $\xi = r$, la fonction a est une fonction d'allocation linéaire pour (D, Θ) et t.
(ii) si ξ est non colinéaire à r et $\lambda^T \xi \neq 0$, considérons $P = \lceil (|\lambda^T M \xi| + 1)/\lambda^T r \rceil$ et $P = p \times a(r)$ où
— $\lceil u \rceil$ représente le plus petit entier supérieur ou égal à u.
— M est tout entier tel que $\forall x \in D, \forall n > M, x + n \times \xi \in D$.

La fonction a' définie par

$$\begin{array}{rccc} a' : & \mathbf{Z}^n & \rightarrow & \mathbf{Z}^{n-1} \\ & x & \rightarrow & a(x) \bmod P \end{array}$$

est une fonction d'allocation quali-linéaire pour (D, Θ) et t.

Un deuxième théorème définit de même une fonction d'allocation quasi-linéaire dans le cas où la fonction de temps est quasi-affine.

A chaque point de $a(D)$ correspond une cellule du réseau. Chaque cellule a un port d'entrée noté $I(u_i)$ et un port de sortie $O(u_i)$ associés à chaque variable u_i définie dans le système d'équations récurrentes uniformes (3). Le port $I(u_i)$ de la cellule Π est connecté au port $O(u_i)$ de la cellule $\Pi + a(\theta_i)$ tandis que le port $O(u_i)$ de la cellule Π est connecté au port $I(u_i)$ de la cellule $\Pi - a(\theta_i)$. Le temps de communication entre deux ports associés est de $t(\theta_i)$ unités de temps.

Pour le produit de convolution décrit par le système (4), la fonction d'allocation associée au vecteur $\xi = (1,0)$ est linéaire et définie par $a(i,k) = k$. Le réseau correspondant est donc constitué de $K+1$ cellules où les données w_k sont constantes des cellules $(a(\theta_w) = 0)$ et où les données x_i et y_i circulent de gauche à droite $(a(\theta_x) = a(\theta_y) = -1)$ avec un retard sur le flot des x_i car $t(\theta_x) = -2$ comme le montre la figure 5.3 pour $K = 2$.

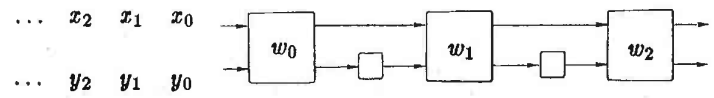


FIGURE 5.3. Exemple de réseau systolique

Ainsi la méthode proposée par QUINTON permet, comme celle proposée dans cette thèse, de concevoir complètement et automatiquement une solution systolique d'un problème à partir d'une spécification en termes de système d'équations récurrentes uniformes et d'un choix pour les fonctions t et a. L'objectif de Quinton est également d'utiliser cette méthode dans un système de conception assistée d'architectures systoliques appelé DIASTOL (*cf.* [QUG 84]).

Une étude comparative des deux méthodes conduit aux constatations suivantes.

La notion de domaine de dépendance est la même dans les deux méthodes et représente les indices des différents calculs élémentaires. La décomposition en modules abstraits proposée comme point de départ de notre méthode conduit à des domaines convexes possédant au plus un rayon (seule la première décomposition peut être infinie). Ceci correspond aux restrictions proposées par Quinton qui ne donne de théorème constructif de détermination de la fonction de temps que dans le cas où le domaine est convexe.

La notion de vecteur de dépendance de Quinton et celle de vecteur générateur de notre approche sont similaires au sens où elles définissent les calculs utilisant la même donnée. Elles diffèrent néanmoins pour la raison suivante.

Un vecteur de dépendance implique une relation temporelle entre les calculs : soit θ un tel vecteur. Considérons un point x de D, le calcul du point $y = x + \theta$ de D devra nécessairement s'exécuter après le calcul du point x car la fonction de temps doit vérifier $t(x) < t(y)$. A partir d'un système d'équations récurrentes uniformes, la méthode de Quinton permet donc d'obtenir une "famille" de réseaux systoliques ayant les mêmes contraintes temporelles entre les calculs dépendants. D'où la nécessité de pouvoir déterminer d'autres systèmes d'équations équivalents qui permettront de déterminer de nouvelles "familles" de réseaux systoliques.

Notre notion de vecteur générateur, par contre, n'impose pas de contrainte temporelle entre les calculs dépendants. Selon la suite de transformations appliquée, ces calculs pourront être ordonnés dans un sens ou dans l'autre, voir même simultanés. A partir de la spécification du problème, la méthode permet donc dans ce cas de déterminer plusieurs "familles" différentes de réseaux.

Pour définir l'ordre des calculs, QUINTON détermine, en fonction du graphe de dépendance (D, Θ), une fonction de temps $t = (\lambda, \alpha)$, affine ou quasi-affine. Notons que le théorème proposé donne toutes les valeurs possibles de λ et α. Il peut y en avoir une infinité. Un système de conception automatique ne saurait traiter une infinité de fonctions de temps. Il doit se limiter aux fonctions de temps "intéressantes". Or l'article de QUINTON ne précise pas explicitement quelles sont les valeurs "intéressantes" de λ et α.

Dans notre méthode, l'ordonnancement temporel des calculs est imposé par l'application d'une suite de transformations affines. Ces transformations correspondent aux fonctions de temps affines de Quinton. Pour avoir une correspondance avec les fonctions de temps quasi-affines de Quinton, il faudrait définir dans notre méthode des transformations quasi-affines. Elles permettraient d'obtenir de nouveaux réseaux comme celui de la figure 5.4 obtenu par Quinton avec la fonction de temps $t(i,k) = \lfloor \frac{i}{2} + k \rfloor$ appliquée au système d'équations (5). La transformation quasi-affine correspondante est représentée à la figure 5.5.

La notion de fonction d'allocation est la même dans les deux méthodes. Notons que dans le cas d'un domaine infini, la direction d'allocation dans notre méthode est toujours parallèle au rayon du domaine (fonction d'allocation linéaire). Quinton propose une solution plus générale avec des fonctions quasi-linéaires qui conduisent à des réseaux en anneaux.

Lorsque le domaine est infini et que la direction d'allocation choisie n'est pas parallèle au rayon du domaine, la fonction linéaire $a(x), x \in \mathbf{Z}^n$ obtenue, ne peut être considérée comme la fonction d'allocation, puisqu'elle ne projette pas le domaine sur un ensemble fini de cellules. La fonction d'allocation est alors définie par $a'(x) = a(x) \bmod P$ où P est une constante calculée à partir du rayon. Une telle fonction est quasi-linéaire.

Une extension possible de notre proposition pourrait aller dans ce sens s'il s'avère que de tels réseaux sont intéressants dans la pratique.

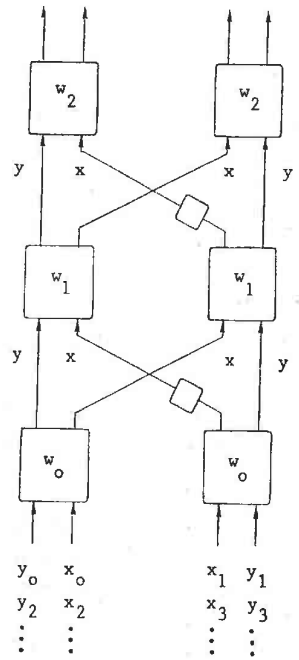


FIGURE 5.4. Réseau systolique correspondant à $t = \lfloor \frac{i}{2} + k \rfloor$

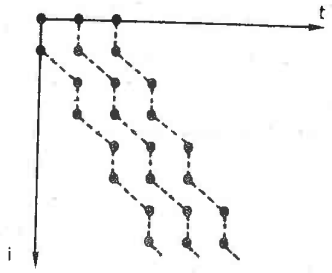


FIGURE 5.5. Exemple de transformation quasi-affine

Le choix de Quinton de définir un problème par un système d'équations récurrentes uniformes nécessite de disposer de transformations formelles permettant de définir des systèmes équivalents pour obtenir un ensemble important de réseaux différents. Ceci sera résolu, dans le système de conception assistée DIASTOL par l'utilisation d'un système de règles de réécriture permettant de déduire des systèmes d'équations récurrentes uniformes équivalents (*cf.* [QUG 84]).

Notre approche, par contre, permet d'obtenir des réseaux différents à partir d'une spécification unique. L'ensemble des réseaux obtenus est réduit aux réseaux classiques en raison des restrictions mentionnées ci-dessus sur les transformations et sur le choix de la direction d'allocation ξ dans le cas d'un domaine infini.

V. Comparaison des différentes approches

Nous pouvons résumer les similitudes et les différences entre ces méthodes et celle proposée dans cette thèse par le tableau 6.1, en nous appuyant sur les critères proposés au paragraphe I.

Les points de départ de chacune des méthodes sont relativement voisins. Néammoins, il semble artificiel de partir d'un algorithme séquentiel plutôt que d'une définition par des équations.

Les trois méthodes proposées se limitent aux réseaux systoliques purs. De ce fait, les auteurs souhaitent imposer une absence de diffusions dès la définition du problème. Moldovan et Miranker-Winkler le réalisent en transformant l'algorithme par l'introduction de tous les indices de boucles dans les calculs. Notons qu'ils n'indiquent pas comment cela peut être réalisé de manière systématique. Quinton le réalise par le biais d'un système d'équations récurrentes uniformes, qui impose des dépendances entre les calculs. De ce fait, les réseaux obtenus appartiennent tous à une même famille où les dépendances entre les calculs sont les mêmes. Pour pallier à ce fait, il faut donc pouvoir transformer un système d'équations récurrentes uniformes en un autre système équivalent, qui impose d'autres dépendances de données et permet ainsi d'obtenir d'autres familles de réseaux systoliques. Dans l'état actuel des propositions de Quinton, ce passage qui doit être réalisé par un système de réécriture n'est pas décrit en détail.

Critères \ Méthode	Moldovan	Miranker-Winkler	Quinton	Systol
Point de départ	Algorithme séquentiel (boucles)	Algorithme séquentiel (boucles)	Equations récurrentes uniformes	Equations récurrentes
Caractérisation des calculs élémentaires	Ensemble $\mathcal{J}^n$	Ensemble Γ	Domaine D	Domaine D
Caractérisation des calculs utilisant la même occurence de donnée	Vecteur de dépendance	Vecteur de dépendance	Vecteur de dépendance	Vecteur générateur
Définition des calculs simultanés	fonction Π	1ère partie de la transformation définie sur la matrice des vecteurs de dépendance	fonction de temps	abscisse temporelle des points pour une représentation obtenue par application de transformations sur la représentation initiale
Définition de la géométrie du réseau	fonction S	2ème partie de la transformation définie sur la matrice des vecteurs de dépendance	fonction d'allocation	fonction d'allocation
Méthode constructive	non	non	oui	oui
Logiciel réalisable*	non	non	oui	oui

* dans l'état actuel de la méthode

Tableau 6.1

Dans toutes les méthodes proposées, la caractérisation des calculs élémentaires par l'ensemble de leurs indices est la même. La notion de vecteur de dépendance ou de vecteur générateur permet, dans toutes les méthodes, de définir les calculs utilisant la même occurence d'une donnée. Ils sont déduits du point de départ du problème (algorithme ou équations).

L'objectif des différentes méthodes est le même : définir, pour chaque calcul élémentaire,
— son instant d'exécution,
— son endroit d'exécution, c'est-à-dire la référence de la cellule du réseau où il est exécuté.

Pour réaliser cela, Moldovan et Miranker-Winkler appliquent à la matrice des vecteurs de dépendance une transformation telle que les vecteurs de dépendance images vérifient certaines conditions. Ces conditions traduisent le fait que l'ordonnancement des calculs obtenus est compatible avec leurs dépendances. La transformation est partitionnée en deux fonctions, l'une correspondant au temps, qui définit pour chaque calcul son instant d'exécution, l'autre qui définit la cellule où il est réalisé.

Quinton définit l'instant d'exécution de chaque calcul élémentaire en déterminant une fonction de temps. Cette fonction est affine ou quasi-affine et doit vérifier une condition sur les vecteurs de dépendance qui garantit un ordonnancement correct des calculs. La cellule associée à chaque calcul est définie par une fonction d'allocation linéaire ou quasi-linéaire.

Dans la méthode proposée ici, l'instant d'exécution de chaque calcul est définit par son abscisse temporelle sur une représentation considérée, obtenue par application d'une suite de transformations affines à partir de la représentation initiale du problème. La cellule associée à un calcul est aussi définie par une fonction d'allocation linéaire.

Les méthodes de Moldovan et Miranker-Winkler ne sont pas des méthodes constructives et ne permettent donc pas de réaliser un logiciel de construction automatique de réseaux systoliques.

Nous comparons donc la méthode de Quinton et celle proposée ici. La figure 6.2 résume quels sont les réseaux que chacun des logiciels correspondants permet de déterminer.

Notons qu'une extension de notre méthode à des fonctions de temps quasi-affines et à des fonctions d'allocation quasi-linéaires (*cf.* paragraphe

DIASTOL	SYSTOL
réseaux d'une même famille (1)	réseaux de familles différentes
	réseaux systoliques avec diffusion
réseaux systoliques purs - de forme classique (2) - en anneaux - avec double cheminement des données (*cf.* fig. 6.4)	réseaux systoliques purs - de forme classique

(1) ensemble de réseaux avec les mêmes dépendances entre les calculs.
(2) flots linéaires

FIGURE 6.2 Comparaison des possibilités de DIASTOL et SYSTOL

IV), permettraient d'obtenir les réseaux particuliers avec double cheminement des données ou en anneaux.

L'intérêt de notre méthode est qu'elle permet de construire un système d'aide à la conception de réseaux systoliques SYSTOL. Ce logiciel détermine, pour une définition d'un problème, des réseaux semi-systoliques et systoliques purs de différentes familles.

CONCLUSION

Notre objectif était de proposer une méthode constructive de conception d'algorithmes systoliques et de la réaliser dans un logiciel.

Cette méthode détermine, à partir de la spécification systolique d'un problème, un ensemble de représentations ou ordonnancements des calculs élémentaires et calcule, pour chaque représentation, un ensemble de réseaux, semi-systoliques avec diffusion ou systoliques purs, solutions de ce problème.

Elle a été mise en œuvre dans un logiciel appelé SYSTOL qui détermine les réseaux systoliques associés à un problème , les affiche sur un écran bit-map, et donne la possibilité à l'utilisateur de visualiser une simulation d'exécution, par déplacement pas à pas des données sur les flots.

Notons que la méthode se distingue des autres méthodes étudiées essentiellement par le non rejet systématique de la diffusion de données et par les outils et méchanismes utilisés pour définir l'ordonnancement temporel des calculs élémentaires. Elle doit être approfondie pour permettre de résoudre des problèmes plus complexes.

Le logiciel réalisé actuellement est un prototype sur lequel il y a encore beaucoup à faire pour le rendre plus facilement utilisable et plus performant, comme nous l'avons indiqué au chapitre 5.

Cette thèse n'est qu'une étape dans le travail qui peut être envisagé. Outre un approfondissement et une extension de la méthode, des pistes de recherche se dessinent à la fois en "amont" et en "aval" de la méthode proposée.

En amont de la méthode, une étude théorique plus approfondie pourra être menée sur le processus de décomposition en modules abstraits présenté au chapitre 5 I.2. Cela pourrait permettre de ne plus définir le problème par un système d'équations qui n'est pas toujours simple à déterminer et d'obtenir directement, à partir de cette décomposition, la spécification systolique du problème.

Au niveau de la méthode, des extensions sont envisageables dans les directions suivantes :
— calculer les directions d'allocation $\vec{\xi}$ associées au type de réseaux souhaités.

— considérer des transformations quasi-affines et des fonctions d'allocation quasi-linéaires ainsi que nous l'avons évoqué au chapitre 6.
— approfondir la notion de performances de réseaux.
— traiter le cas des communications (lorsqu'un résultat sert de donnée dans le calcul d'un autre résultat élémentaire).

En aval de la méthode, sont à étudier tous les problèmes liés à la réalisation physique des réseaux systoliques obtenus.

ANNEXE

Nous présentons dans cette annexe quelques uns des programmes du logiciel :

— le module DEFIN ou systol-definition, qui décrit les structures de données du problème.
— le module SYSMAIN, qui contient le programme principal (PROGRAM systolique).
— le module TRANSFO ou systol-transfo qui définit l'ensemble des procédures nécessaires à la création de l'arbre des représentations.
— une partie du module RESEAU qui permet la construction des réseaux associés à une représentation donnée.

```
MODULE systol_definition;

CONST infini  = 9999;
      max_vg  = 20;        { nombre maximum de vecteurs generateurs =
                             nombre maximum de familles de ressources + 1
                             car resultat peut communiquer ==> 2 vect. gen }
      max_ssd = 2;         { nombre maximum de sous-domaines = ens. des
                             points utilisant la meme fonction et les memes
                             ressources D=D1 U D2}
      max_car = 15;        { nombre maximum de caracteres dans une chaine }
      hauteur = 780;       { nombre de points de la fenetre NUMELEC }
      longueur= 1024;      { nombre de points de la fenetre NUMELEC }
      bord    = 100;       { nombre de points laisses pour les bords }

TYPE  vecteur        = ARRAY [1..3] OF integer;
      t2_3           = 2..3;
      chaine         = PACKED ARRAY [1..max_car] OF char;
      nat_symb       = (cte,nom,op);
      p_expr         = ^expr;
      expr           = RECORD
                         suiv : p_expr;
                         CASE ns : nat_symb OF
                          cte    : (ent : integer);
                          nom    : (id : 'i'..'k');
                          op     : (oper : char);
                        END;
                        { expr pour memoriser les expresions arithemiques
                          definissant les fonctions, les bornes de dommaine
                          sous forme polonaise}

      couple         = RECORD
                         inf,sup : p_expr;
                       END;
                       { pour memoriser les bornes inf et sup sur une composante
                         i,j ou k. les bornes peuvent etre constantes (=infini) }

      repres         = RECORD
                         b : ARRAY [1..3] OF integer;
                         v : integer;
                       END;
                       { une representation est caracterisee par la matrice
                         de passage P de vecteurs colonnes Bi = (1,0,b1),
                         Bj = (0,1,b2) et Bk = (0,0,b3) exprimes dans l'espace
                         temps et par le vecteur de translation V = (0,0,v) }

      t_comptvg      = ARRAY [0..max_vg] OF integer;
      arbre          = RECORD
                         rep       : repres;
                         interm    : boolean;   {a vrai si rep intermediaire
                                                 entre rep init et rep depart }
                         equiv     : boolean;   {a vrai si rep equivalente a
                                                 rep precedente }
                         comp_t_vg : t_comptvg; {valeur de la composante sur t
                                                 de chaque vg }
                         TcOi,TdOi : ^arbre;
                         TcOj,TdOj : ^arbre;
```



```
                         Tecart    : ^arbre;
                       END;

      lien           = ^arbre;       { arbre des differentes representations }

      fonct_ress     = RECORD
                         util  : boolean;  { a vrai si ressource utilisee }
                         f1,f2 : p_expr; {f2 = nil si suite simple
                                          f1 et f2 = nil si donnee simple }
                         f1min,f1max,f2min,f2max : integer;
                                  { valeurs min et max de f1 et f2 sur
                                    le sous-domaine concerne }
                         f1cte,f2cte : integer;  { les fonctions f1 et f2 sont
                                    lineaires de la forme a1+bj+ck+cte }
                       END;
      t_fress        = ARRAY [0..max_vg] OF fonct_ress;
      t_vectgen      = ARRAY [0..max_vg] OF vecteur;
      t_bornes       = ARRAY ['i'..'k'] OF couple;
      sous_domaine   = RECORD
                         bornes    : t_bornes;
                         fonction  : p_expr;
                         def_ress  : t_fress;
                       END;
                       { fonction utilisee en tous points du sous-domaine

                         ress (1) expression def. comment est utilisee la
                         ieme resource sur le sous-domaine. si elle n'est
                         pas utilisee, ress (1) = nil.
                         ress (0) definit le resultat utilise comme donne
                         quand communication }

      ressource      = RECORD
                         dim : 1..3;
                         nom : char;
                       END;
      tab_ressource  = ARRAY [0..max_vg] OF ressource;
      specif_syst    = RECORD
                         dimension     : t2_3;
                         born_dom      : t_bornes;
                         nbr_vg        : integer;
                         ress          : tab_ressource;
                         vect_gen      : t_vectgen;
                         nbr_ssd       : integer;
                         ss_dom        : ARRAY [1..max_ssd] OF sous_domaine;
                         communic      : boolean;     { a vrai si communication}
                         contrainte    : boolean;     { a vrai si contrainte init }
                       END;
                       { dimension = 2 si pb lineaire, = 3 si pb bidimens.
                         vect-gen exprimes dans la base canonique du reseau
                         vect-gen [0] definit res. utilise comme donnee qd
                         communication
                         ressources = liste des noms de ressources reduits
                         a 1 caractere ressources [1] = resultat }
VAR spesys     : specif_syst;
    arbo       : lien;
    pb_infini  : boolean;       { a vrai si le domaine est infini sur i }
```

```
PROGRAM systolique (input,output);
USE sysdefin, traitexpr, saisie, systransfo, verif, graphic, reseau;

{ sysdefin decrit les donnees du probleme et definit deux variables globales
          spesys : specif-syst decrivant la caracterisation du probleme
          arbo : lien pointeur sur la racine de l'arbre des representations }

{ saisie lit les donnees de la caracterisation du probleme : spesys }

{ systransfo construit l'arbre des representations du probleme }

{ verif definit les procedures de verif provisoires }

{ genere les reseaux }

VAR ch : char;

BEGIN
 saisie;
 imprime (spesys);   (***** verif de saisie *****)
 construction (arbo);
 init_parcours; parcours (arbo,1);  (***** verif de creation d'arbre *****)

{ chargement de la fonte }
 charge_fonte (font);

(******  construction des reseaux ********)
 init_file (fil_rep);
 creat_reseau (arbo);
 REPEAT
  writeln; writeln;
  write ('    **** Voulez-vous revoir les differents reseaux (o/n)? '); flush (output);
  readln; read (ch);
  IF ch IN ['o','O'] THEN
  BEGIN
   init_file (fil_rep);
   creat_reseau (arbo);
  END
 UNTIL NOT (ch IN ['o','O']);
END.
```



```
MODULE systol_transfo;
USE sysdefin, traitexpr;
VAR rep_init : boolean;

PROCEDURE construction (VAR arbo : lien);
{ construction de l'arbre des differentes representations }

TYPE t_liste = RECORD
                r : repres;
                s : ^t_liste;
               END;
     p_liste = ^t_liste;
     info    = RECORD
                Ti,Tj : ARRAY ['c'..'d'] OF boolean;
                Te    : boolean;
               END;
VAR inf_i,sup_i,inf_j,sup_j,vk : integer;
    liste                      : p_liste;
    PremOi,PremOj              : char;
    don                        : info;

     { Ti['c'] = booleen indiquant si une transfo TcOi (du type 1) est applicable,
           a vrai si TcOi applicable (c-a-d si un vecteur generateur a sa
             composante sur t (comp-t) nulle et sa composante sur i non nulle
           a faux sinon . idem pour Ti['d'],Tj['c'],Tj['d'] }
     { Te = booleen indiquant si une transfo du type 5 (ecartement des hyperplans)
           est applicable,
           a vrai si applicable (c-a-d si un vecteur generateur a sa composante
             sur t nulle et sa composante sur k non nulle }

 PROCEDURE calcul_comp_t (r : repres; VAR t : t_comptvg);
 { calcul la composante sur t dans la representation r de chaque vecteur
   generateur et la memorise dans le tableau t }

 VAR i,j,min : integer;
 BEGIN
  WITH spesys DO
  BEGIN
   IF communic THEN min := 0
   ELSE min := 1;
   { si communication le vg [0] definit le resultat utilise comme donnee }
   FOR i := min TO nbr_vg DO
   BEGIN
    t[i] := 0;
    FOR j := 1 TO 3 DO t [i] := t [i] + vect_gen [i,j] * r.b [j];
   END;
  END;
 END;    { calcul_comp_t }

 PROCEDURE transfo_applic (VAR p : lien; VAR PremOi,PremOj : char; VAR don : info);

 { determination des transfos applicables sur la representation pointee par p }
 { utilisation d'une variable globale rep_init de type booleenne qui indique
   si la representation est initiale ou non }
 { PremOi definit la premiere transformation sur Oi : c pour Tc, d pour Td, v
   pour aucune }
```

```
VAR comp_t,i,j     : integer;

    { comp_t = composante sur t dans l'espace temps d'un vecteur generateur vg
             = 3eme ligne de P (b1,b2,b3) * vg (base canonique).
               les composantes sur i et j sont les memes que sur Bi et Bj }

BEGIN
 { init a faux des booleans }
 don.Ti ['c'] := false; don.Ti ['d'] := false;
 don.Tj ['c'] := false; don.Tj ['d'] := false;
 don.Te := false;

 { examen de chaque vecteur generateur. si sa composante sur t est nulle,
   certaines transfos sont applicables }
 FOR i := 1 TO spesys. nbr_vg DO
 BEGIN
  { traitement du ieme vecteur generateur. Determination de sa comp sur t }
  comp_t := 0;
  FOR j := 1 TO 3 DO comp_t := comp_t + p^.rep.b [j] * spesys.vect_gen [i,j];

  IF comp_t = 0 THEN      { transfos applicables }
  BEGIN
   IF spesys.vect_gen [i,1] <> 0 THEN  { comp. sur i <> 0 ==> T(Oi) applic. }
   BEGIN
    CASE PremOi OF
     'c': don.Ti ['c'] := true;
     'd': don.Ti ['d'] := true;
     'v': BEGIN
           don.Ti ['c'] := true;
           IF NOT pb_infini THEN don.Ti ['d'] := true;
          END;
    END;
   END;
   IF spesys.dimension = 3 THEN      { transfo sur J possibles }
    IF spesys.vect_gen [i,2] <> 0 THEN   { comp. sur J <> 0 ==> T(Oj) applic. }
    BEGIN
     CASE PremOj OF
      'c': don.Tj ['c'] := true;
      'd': don.Tj ['d'] := true;
      'v': BEGIN
            don.Tj ['c'] := true;
            don.Tj ['d'] := true;
           END;
     END;
    END;
   IF spesys.vect_gen [i,3] <> 0 THEN      { comp. sur k <> 0 ==> don.Tecart app. }
    don.Te := true;
  END;
 END {FOR};

END  {transfo-applic };


PROCEDURE ajout_liste (VAR l : p_liste; VAR p : lien);
{ insertion en tete de la liste des representations non equivalentes pointee
```



```
  par l de la representation pointee par p dans l'arbre }
VAR cour : p_liste;
BEGIN
 new (cour);
 cour^.r := p^.rep;
 cour^.s := l;
 l := cour;
END;     { ajout-liste }

FUNCTION rep_interm (VAR p : lien): boolean;
BEGIN
 rep_interm := false;    (********************* A FAIRE *********************)
END;     { rep-interm }

FUNCTION rep_equiv (VAR l : p_liste; VAR p : lien): boolean;
{ determine si la representation pointee par p dans l'arbre est ou non
  equivalente a une representation precedemment obtenue, memorisee dans
  la liste pointee par l }
VAR cour : p_liste;
    trouve : boolean;
    bp1,bp2,bp3 : integer;
BEGIN
 bp1 := p^.rep.b[1];
 bp2 := p^.rep.b[2];
 bp3 := p^.rep.b[3];
 trouve := false;
 cour := l;
 WHILE (cour <> nil) AND NOT trouve DO
 BEGIN
  WITH cour^.r DO
   trouve := (b[3]*bp1 = b[1]*bp3) AND (b[3]*bp2 = b[2]*bp3);
  cour := cour^.s;
 END;
 rep_equiv := trouve;
END;   { rep-equiv }

PROCEDURE insertion (VAR p,q : lien; sens,ref : char; PremOi,PremOj : charid:info);
{ insertion dans l'arbre en q d'une representation dont les caracteristiques
  dependent de la representation pere (pointee par p) et du sens de la
  transformation appliquee
   - c ou d ainsi que de l'axe de reference (i ou j)
   - e pour la transfo de type 5 (sens sans importance) }
VAR aux : lien;
BEGIN
 new (aux);
 WITH aux^ DO
 BEGIN
  rep := p^.rep;
  WITH rep DO
  BEGIN
   CASE sens OF
    'c': CASE ref OF
          'i': b[1] := b[1] + 1;
          'j': b[2] := b[2] + 1;
         END;
    'd': CASE ref OF
```

```
          'i': BEGIN
                 b[1] := b[1] - 1;
                 v := v + sup_i -inf_i;
               END;
          'j': BEGIN
                 b[2] := b[2] - 1;
                 v := v + sup_j - inf_j;
               END;
        END;
      'e': b[3] := b[3] + 1;
     END;
    END;  {WITH}
    { maj de PremOi et PremOj }
    IF (PremOi = 'v') AND (ref = 'i') THEN PremOi := sens;
    IF (PremOj = 'v') AND (ref = 'j') THEN PremOj := sens;

    interm := rep_interm (aux);
    IF NOT interm THEN
    BEGIN
     equiv := rep_equiv (liste,aux);
     IF NOT equiv THEN ajout_liste (liste,aux);
    END;
    calcul_comp_t (rep,comp_t_vg);
   END;   {WITH}
   transfo_applic (aux,PremOi,PremOj,d);
   q := aux;
   IF  NOT d.Ti ['c']  THEN q^.TcOi := nil
   ELSE insertion (q,q^.tcOi,'c','i',PremOi,PremOj,d);
   IF NOT d.Ti ['d']  THEN q^.TdOi := nil
   ELSE insertion (q,q^.TdOi,'d','i',PremOi,PremOj,d);
   IF NOT d.Tj ['c']  THEN q^.TcOj := nil
   ELSE insertion (q,q^.TcOj,'c','j',PremOi,PremOj,d);
   IF NOT d.Tj ['d']  THEN q^.TdOj := nil
   ELSE insertion (q,q^.TdOj,'d','j',PremOi,PremOj,d);
   IF NOT d.Te THEN q^.Tecart := nil
   ELSE insertion (q,q^.Tecart,'e','e',PremOi,PremOj,d);
  END;  { Insertion }

BEGIN   { construction }
 { determination des valeurs des bornes inf et sup sur i et j pour le calcul
   de v quand les Td sont possibles }
 WITH spesys DO
 BEGIN
  inf_i := born_dom ['i'].inf^.ent;      { sur i, bornes constantes ...   }
  sup_i := born_dom ['i'].sup^.ent;      { ==> ...^.suiv = nil }
  IF dimension = 2 THEN
  BEGIN                                  { transfo sur j impossible }
   inf_j := 0;
   sup_j := 0;
  END
  ELSE   { dimension = 3 }
  BEGIN
   { evaluation de la borne inferieure sur j. Cette borne peut dependre de
     i et peut-etre de k ( si depend = true ) }
   IF dependance (born_dom ['j'].inf,'k') THEN
   BEGIN
```



```
    eval_expr (born_dom ['k'].inf,born_dom['i'].inf^.ent,0,0,vk);
    eval_expr (born_dom ['j'].inf,born_dom['i'].inf^.ent,0,vk,inf_j);
   END
   ELSE eval_expr (born_dom ['j'].inf,born_dom['i'].inf^.ent,0,0,inf_j);
   { evaluation de la borne superieure sur j. Cette borne peut dependre de
     i et peut-etre de k (si dependance) }
   IF dependance (born_dom['j'].sup,'k') THEN
   BEGIN
    eval_expr (born_dom ['k'].sup,born_dom['i'].sup^.ent,0,0,vk);
    eval_expr (born_dom ['j'].sup,born_dom['i'].sup^.ent,0,vk,sup_j);
   END
   ELSE eval_expr (born_dom ['j'].sup,born_dom['i'].sup^.ent,0,0,sup_j);
  END;
 END;

 { init de la racine arbo avec la representation initiale }
 new (arbo);
 WITH arbo^ DO
 BEGIN
  WITH rep DO
  BEGIN
   b[1] := 0;
   b[2] := 0;
   b[3] := 1;
   v := 0;
  END;
  equiv := false;
  interm := rep_interm (arbo);
  calcul_comp_t (rep,comp_t_vg);
  liste := nil;   { init de la liste des representations non equivalentes }
  IF NOT interm THEN ajout_liste (liste,arbo);
  { pour la representation initiale, PremOi et PremOj sont vides }
  PremOi := 'v'; PremOj := 'v';

  transfo_applic (arbo,PremOi,PremOj,don);
  IF NOT don.Ti ['c']  THEN TcOi := nil
  ELSE insertion (arbo,arbo^.TcOi,'c','i',PremOi,PremOj,don);
  IF NOT don.Ti ['d']  THEN TdOi := nil
  ELSE insertion (arbo,arbo^.TdOi,'d','i',PremOi,PremOj,don);
  IF NOT don.Tj ['c']  THEN TcOj := nil
  ELSE insertion (arbo,arbo^.TcOj,'c','j',PremOi,PremOj,don);
  IF NOT don.Tj ['d']  THEN TdOj := nil
  ELSE insertion (arbo,arbo^.TdOj,'d','j',PremOi,PremOj,don);
 END;
END;.
```

```
MODULE reseau;

USE sysdefin, traitexpr, traitcond, rasteruse, graphic;

PROCEDURE determ_reseau (VAR p : lien);

{ variables locales non explicitees }

BEGIN   { determ-reseau }
 REPEAT
  diralloc (ksi,a);   { lecture de la direction d'allocation et
                        determination de la fonction d'allocation a }
  taille_min := taille_celret * (max_celret * 2 + 1);
  config_cellule (ok);
  IF ok THEN
  BEGIN    { reseau dessinable sur l'ecran }
   config_flot;         { representation du sens des flots et des eventuelles
                          cellules de retard }

   { calcul de l'espacement des donnees sur les flots = ksi [t] - 1
     ou ksi [t] = 3eme ligne de p (b1,b2,b3)  par ksi (dans B.C.) }
   WITH p^.rep DO
    espac := abs (ksi[1] * b[1] + ksi[2] * b[2] + ksi[3] * b[3]) - 1;

   t := 0;       { t = instant de simulation }
   config_don (t,vid_str);      { representation des donnees sur le reseau }
                                { a l'instant t }

   { copie d'ecran du reseau dessine }
   writeln;
   write ('   Voulez-vous imprimer ce reseau sur listing (o/n) ? ');
   flush (output);
   readln; read (ch);
   IF ch IN ['o','O'] THEN copy_ecran;

   writeln;
   write ('   Simulation du reseau. A chaque demande s''affiche l''etat du ');
   writeln ('reseau en t,t+1..');
   writeln;
   write ('   Voulez-vous visualiser la simulation (o/n) : '); flush (output);
   readln; read (ch);
   WHILE NOT (ch IN ['n','N']) DO
   BEGIN
    config_don (t,vid_not);        { reaffiche les donnees a l'instant t }
                                   { ==> efface }
    t := t + 1;
    config_don (t,vid_str);        { affiche les donnees au nouvel instant }
    write ('   Voulez-vous continuer (o/n) : '); flush (output);
    readln; read (ch);
   END;
  END
  ELSE writeln ('***** Dessin sur ecran impossible: trop de cellules ****');
  writeln;
  write ('   Autre direction d''allocation (o/n) : '); flush (output);
  readln; read (ch);
UNTIL ch IN ['n','N'];
```



```
END;   { determ_reseau }

PROCEDURE creat_reseau (VAR p : lien);
BEGIN
 IF (NOT p^.interm) AND (NOT p^.equiv) THEN
 BEGIN
  ecrire_file (fil_rep);
  determ_reseau (p);            { recherche des reseaux correspondants }
 END;
 IF p^.TcOi <> nil THEN
 BEGIN
  enfiler (fil_rep,'TcOi');
  creat_reseau (p^.TcOi);
  defiler (fil_rep);
 END;
 IF p^.TdOi <> nil THEN
 BEGIN
  enfiler (fil_rep,'TdOi');
  creat_reseau (p^.TdOi);
  defiler (fil_rep);
 END;
 IF p^.TcOj <> nil THEN
 BEGIN
  enfiler (fil_rep,'TcOj');
  creat_reseau (p^.TcOj);
  defiler (fil_rep);
 END;
 IF p^.TdOj <> nil THEN
 BEGIN
  enfiler (fil_rep,'TdOj');
  creat_reseau (p^.TdOj);
  defiler (fil_rep);
 END;
 IF p^.tecart <> nil THEN
 BEGIN
  enfiler (fil_rep,'Tec');
  creat_reseau (p^.Tecart);
  defiler (fil_rep);
 END;
END;
```

BIBLIOGRAPHIE

[AFC 83] Calcul vectoriel et parallèle. Actes du colloque AFCET-GAMNI-ISINA. 17-18 mars 1983, Paris.

[AFQ 83] ANDRE F., FRISON P., QUINTON P., Algorithmes systoliques : de la théorie à la pratique. IRISA Publication interne 196, mars 1983.

[BAR-al 68] BARNES G.H., BROWN R.M., KATO M., KUCK D.J., SLOTNICK D.L., STOKES R.A., The ILLIAC IV Computer. IEEE Transactions in computers, C 17, p. 746-757. 1968.

[BER-al 81] BERGER P., COMTE D., HIFDI N., PELOIS B., SYRE J.C., Le système LAU : multiprocesseur à assignation unique. TSI vol 1. No 1, 1981.

[BLA 63] BLANKINSHIP W.A., A new version of the Euclidean algorithm. American Mathematical Monthly, p 742-745, septembre 1963.

[BOM 81] BOSSAVIT A., MEYER B., The design of vector programs. Proc. International Conference on Algorithmic Langages, Nth Holland, Amsterdam, 1981.

[BOS 81 a] BOSSAVIT A., La vectorisation de Cholesky. EDF Bulletin de liaison de la direction des Etudes et Recherches. Série C. No 1 1981. p. 5-18.

[BOS 81 b] BOSSAVIT A., Vectorisation de l'algorithme de Gauss-Seidel. EDF Bulletin de liaison de la direction des Etudes et Recherches. Série C. No 2 1981. p. 81-88.

[BOS 82] BOSSAVIT A., L'ordinateur vectoriel et les méthodes numériques du calcul des structures. Comm. aux "2-iemes journées sur les tendances actuelles en calcul des structures". 1-3 février 1982, Sophia-Antipolis.

[**CFQ 84**] CHAROT F., FRISON P., QUINTON P., Systolic architectures for connected speech recognition. INRIA Rapport de recherche 332, septembre 1984.

[**COM 82**] "Data Flow Systems", numéro spécial. Computer vol 15-2, février 1982.

[**COS 83**] COMTE D., SYRE J.C., Les supercalculateurs. La Recherche No 147 Vol 14 p.1084, septembre 1983.

[**ENS 77**] ENSLOW P., Multiprocessor organization, a survey. Computing surveys 9, p.103-129, 1977.

[**FLY 72**] FLYNN M.J., Some Computer Organizations and their effectiveness. IEEE Transactions in Computers. Vol C-21, p. 948-960, septembre 1972.

[**FOK 80**] FOSTER M.J., KUNG H.T., The design of special-purpose VLSI chips. Computer, vol 13-1, janvier 1980, p.26-40.

[**HAY-al 82**] HAYNES L.S., LAU R.L., SIEWIOREK D.P., MIZELL D.W., A Survey of Highly Parallel Computing. Computer, vol 15-1, p. 9-24, janvier 1982.

[**HOA 78**] HOARE C.A.R., Communicating Sequential Process. Comm. ACM, vol 21-8, 1978.

[**HOJ 81**] HOCKNEY R.W., JESSHOPE C.R., Parallel computers. Adam Hilger, 1981.

[**ICH-al 79**] ICHBIAH J. et al., Preliminary Ada Reference Manuel. SIGPLAN Notices, 14-6, Part A, juin 1979.

[**INR 84**] INRIA, Bulletin de liaison sur le calcul parallèle, num. 94, juin 1984.

[**JUL 83**] JULLIAND J., Spécification algébrique de la communication entre processus parallèles. TSI vol 2 No 4, 1983.

[**KUL 80**] KUNG H.T., LEISERSON C.E., Algorithms for VLSI processor arrays. p. 271-292 dans [MEC 80].

[**KUN 79**] KUNG H.T., The structure of parallel algorithms. Advances in Computer, vol 19, Academic Press.

[**KUN 82**] KUNG H.T., Why systolic architectures? Computer, vol 15-1, janvier 1982, p. 37-46.

[**LEN 82**] LENFANT J., Mémoires parallèles et réseaux d'interconnexion. TSI, vol 1, No 2, 1982.

[**MEC 80**] MEAD C., CONWAY L., Introduction to VLSI systems. Addison Wesley 1980.

[**MEY 80**] MEYER B. Un calculateur vectoriel : le Cray-1 et sa programmation. Note EDF HI/3452-01, mai 1980.

[**MIW 82**] MIRANKER W.L., WINKLER A., Spacetime representations of systolic computational structures. IBM Research Report RC 9775, décembre 1982.

[**MOL 83**] MOLDOVAN D.I., On the design of algorithms for VLSI systolic arrays. Proceedings of the IEEE, vol 71-1, janvier 1983, p. 113-120.

[**NEW 72**] NEWMANN M., Integal matrices. Academic Press 1972.

[**PAI 79**] PAIR C., La construction de programmes, RAIRO Informatique, vol 13, No 2, p. 113-137, 1979.

[**QUI 83**] QUINTON P., The systematic design of systolic arrays. IRISA Publication interne 193, avril 1983.

[**QUG 84**] QUINTON P., GACHET P., Manuel d'utilisation de DIASTOL, version préléminaire. IRISA Publication interne 233, aout 1984.

[**RUS 78**] RUSSEL R.M., The Cray-1 Computer System, CACM vol 21, No 1, p. 63-72, janvier 1978.

[**TCM 83**] TCHUENTE M., MELKEMI L., Réseaux systoliques pour le calcul des composantes connexes et la triangularisation des matrices bandes. IMAG Rapport de recherche 366, mars 1983.

[**ZAK 84**] ZAKHAROV V. Parallelism and array processing. IEEE Transactions on Computer, vol C 33, No 1, janvier 1984.

institut
national
polytechnique
de lorraine

Le Président,

AUTORISATION DE SOUTENANCE DE THESE

DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

****

VU LES RAPPORTS ETABLIS PAR :

Messieurs les Professeurs DUFOURD
MOHR

le Président de l'Institut National Polytechnique de Lorraine, autorise:

Madame CAVARELLI-MONGENET Catherine

à soutenir devant l'I.N.P.L. une thèse de DOCTORAT intitulée :

"UNE METHODE DE CONCEPTION D'ALGORITHMES SYSTOLIQUES :

RESULTATS THEORIQUES ET REALISATION"

en vue de l'obtention **du titre de**

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : "INFORMATIQUE"

Fait à NANCY, le 29 Avril 1985

Le Président de l'I.N.P.L.

M. LUCIUS

RÉSUMÉ. — Le travail présenté dans cette thèse se situe dans le cadre de la conception d'algorithmes systoliques.

Les architectures systoliques proposées par H.T. KUNG sont des systèmes spécialisés, caractérisés par une structure planaire régulière de processeurs élémentaires connectés localement, à travers lesquels circulent les données de façon rythmée. Ces caractéristiques facilitent leur réalisation par un circuit VLSI.

La conception "à la main" de tels systèmes n'est pas simple. La méthode proposée répond à cette difficulté en déterminant automatiquement, à partir de l'énoncé d'un problème, un ensemble de réseaux systoliques, solutions de ce problème.

Une architecture systolique est adaptée à la résolution de problèmes pouvant s'exprimer par des systèmes d'équations récurrentes. On peut, de manière simple, déduire de ces équations l'ensemble des informations constituant la "spécification systolique" du problème. Cette spécification sert de point de départ de la méthode.

La méthode présentée dans cette thèse permet de déterminer :

— un ensemble de "représentations"—ou ordonnancements temporels des calculs élémentaires—obtenues par application de transformations affines.

— puis, pour chaque représentation, un ensemble de réseaux. Un tel réseau est obtenu en allouant (avec une fonction d'allocation) l'ensemble des calculs élémentaires à un ensemble fini de processeurs et en déterminant les connexions nécessaires.

Cette méthode est mise en œuvre dans un logiciel appelé SYSTOL, qui, à partir de la spécification systolique d'un problème, détermine un ensemble de réseaux et permet la simulation pas à pas de leur exécution.

La structure de chaque réseau (processeurs et connexions) est visualisée sur un écran bitmap sous une forme schématique du type de celle présentée par KUNG : sa configuration initiale est définie par la disposition des données sur les flots d'entrée des boites représentant les processeurs.

La simulation d'exécution est réalisée par la visualisation du déplacement pas à pas de ces données sur le graphe de connexions.

Outre l'automatisation de la conception, le logiciel permet donc la sélection par l'utilisateur du "meilleur" réseau selon ses propres critères : nombre minimal de processeurs, temps minimal d'exécution, simplicité du graphe de connexion, taux de parallélisme, etc.

Si cette méthode présente des points communs avec d'autres méthodes existantes, elle s'en distingue essentiellement par le non-rejet systématique de la diffusion de données et par les outils utilisés pour définir l'ordonnancement temporel des calculs élémentaires.

MOTS CLES : parallélisme, architectures spécialisées, réseaux systoliques, conception automatique d'algorithmes.