THÈSE de DOCTORAT de l'UNIVERSITÉ PARIS 6

Spécialité :

SYSTÈMES INFORMATIQUES

présentée par

Thomas LEDUC

pour obtenir le grade de

DOCTEUR de l' UNIVERSITÉ PARIS 6

Modélisations par réseaux d'automates cellulaires et simulations parallèles du phénomène de subduction-érosion en tectonique des plaques

Soutenance le 5 juillet 1999, devant le jury composé de :

M. René Alt
M. Joffroy Beauquier
M. Jacques Bourgois
M. Paul Feautrier
M. Jean-Luc Lamotte
M. Jacques Mazoyer

à Sofia,

Remerciements

Je tiens, en premier lieu, à remercier tous les membres du jury :

- René Alt (Professeur à l'Université Paris 6) qui a dirigé l'ensemble de mon travail doctoral, m'a intégré dans son équipe de recherche et a su m'appuyer dans les moments décisifs,
- Jacques Bourgois (Directeur de Recherche CNRS, Université Paris 6) qui m'a initié à la géotectonique et qui est un juge indispensable à cette étude consacrée partiellement à la tectonique des plaques,
- Jean-Luc Lamotte (Maître de Conférences à l'Université Paris 6) qui a co-encadré l'ensemble de ce travail,
- Jacques Mazoyer (Professeur à l'ENS-Lyon) et Paul Feautrier (Professeur à l'Université de Versailles) qui me font l'honneur d'être rapporteurs de cette thèse en une période de fin d'année universitaire toujours très chargée,
- Joffroy Beauquier (Professeur à l'Université d'Orsay) qui a accepté de juger ce travail.

Je tiens aussi à remercier vivement l'ensemble des membres de l'*Assistance Utilisateurs* de l'IDRIS (CNRS, Orsay) et plus particulièrement Isabelle Brugeas, Jean-Philippe Proux et Denis Girou. Leur rôle est essentiel et leur assistance est réellement d'un grand secours pour un "utilisateur" en perdition.

Je profite aussi de cette rubrique pour exprimer ma très grande reconnaissance à Vincent Lefebvre (pour son efficace collaboration lors de l'écriture d'un mini-compilateur qui reste à terminer), Jérôme Olivier Durand-Lose (pour son invitation à commencer par un modèle uni-dimensionnel inspiré du Sand Pile Model), Olivier Heen (pour ses conseils avisés en matière d'implémentation informatique des réseaux d'automates cellulaires) et Michel Leduc, mon père (pour ses nombreuses relectures et plus particulièrement pour ce qui concerne le formalisme "automates cellulaires").

Je veux remercier ici, l'ensemble des membres de l'équipe *Calcul Haute Performance et Validation* pour leur convivialité pendant toutes ces années de thèse (avec une mention spéciale à Fabienne Jezequel et David Thibau). J'associe aussi à ces remerciements les personnels des laboratoires LSV et LMT et plus particulièrement leurs directeurs respectifs MM. Michel Bidoit et Pierre Ladevèze pour les facilités qu'ils m'ont accordées dans l'achèvement de ma thèse.

Enfin, je remercie mon épouse Sofia à qui je dédie ce travail et ma mère pour leur infinie patience, leur attention répétée et la prise en main de mon pot de thèse.

Résumé

Dans cette thèse, nous proposons successivement deux modèles discrets par réseaux d'automates cellulaires, du processus de subduction-érosion en tectonique des plaques, puis présentons les simulations informatiques parallèles correspondantes.

Après une présentation de la tectonique des plaques et des marges convergentes de type II (avec érosion), nous présentons les deux tendances de modélisation existantes, étudions leurs avantages et inconvénients respectifs et montrons l'intérêt de développer une démarche radicalement différente. Nous exposons alors nos hypothèses de travail relativement restrictives et leurs limites, en commençant d'abord par présenter la géométrie d'ensemble du "plan de coupe de modélisation" et sa dynamique, puis en énumérant les phénomènes à reproduire, enfin, en introduisant des échelles de temps et la représentation de l'érosion par une altération (un changement de matière) due au vieillissement.

En ce qui concerne les modélisations plus précisément, nous nous inspirons très fortement du "Sand Pile Model" uni-dimensionnel pour développer notre propre modèle uni-dimensionnel et introduire la notion de réseau d'automates cellulaires fini généralisé. Dans le cas du modèle bi-dimensionnel, partant du même principe, nous cherchons à implémenter un modèle d'avalanches dans un tas de sable représenté par un réseau d'automates cellulaires bi-dimensionnel. Constatant que la multiplication des informations stockées dans la structure même du réseau offre un meilleur rendu-visuel, nous choisissons alors de généraliser cette méthode et abordons la description de notre propre réseau d'automates cellulaires.

Les temps de calcul respectifs de chacune des simulations séquentielles ainsi que le fait que les réseaux d'automates cellulaires constituent un modèle canonique du calcul parallèle à fine granularité, nous incitent à développer des simulations parallèles et à les porter sur des ordinateurs parallèles tels que le Cray T3E et l'Origin 2000. Après avoir exposé la stratégie de décomposition de domaine que nous avons employée (avec équi-répartition de la charge des sous-domaines sur l'ensemble des processeurs et minimisation de la taille des problèmes aux interfaces), nous montrons l'intérêt d'utiliser une bibliothèque d'échanges de messages appropriée dans le cadre d'une décomposition de domaine régulière sur une architecture parallèle à mémoire distribuée.

Les résultats obtenus sont révélateurs (pour la simulation bi-dimensionnelle du moins) de la très bonne parallélisabilité du problème posé. Ils nous permettent de présenter quelques copies d'écran des animations graphiques obtenues et leur validation d'un point de vue géotectonique. Des développements futurs pourraient être orientés vers la mise au point d'une plate-forme logicielle parallèle adaptée, puis vers une étude de qualification de la concentration des déformations au sein de la plaque chevauchante.

Abstract

In this thesis, we successively propose two discrete models using cellular automata networks, of the process of subduction-erosion in plate tectonics, then we present the corresponding parallel computing simulations.

After a presentation of the plate tectonics and of convergent margins in subduction with extension, we expose the two existing sorts of models, study their respective advantages and disadvantages and show the interest to develop a radically different one. We describe our relatively restrictive working hypotheses and their limits : first by presenting the overall geometry of the vertical section of the model and its dynamics, then by enumerating the phenomena to be reproduced, at last the times scales and the implementation of erosion by a degradation due to ageing.

Concerning the models, we draw our inspiration from the one-dimensional Sand Pile Model to develop our own one-dimensional model and define finite generalized cellular automata networks. In the case of the two-dimensional model, according to the same principle, we first implement a model of avalanches in a sand pile represented by a two-dimensional cellular automata network. Since the multiplication of the data stored in the structure offers better visual results, we then choose to generalize this method and undertake the description of our own cellular automata network.

The respective computing times of each sequential simulation, as well as the fact that cellular automata networks are classical models of parallel computation with fine granularity, encourage us to develop parallel simulations and to implement them on parallel architectures such as the Cray t3e and the Origin 2000. We expose our domain decomposition strategy and we show what can be gained by using a suitable message-passing library in the field of regular domain decomposition onto parallel architecture.

The results obtained show (for the two-dimensional simulation at least) the very good parallelisability of the problem. We presents some screen dumps of the motion pictures we obtained and their validation from a geotectonics point of view. Future research works could concern the development of a specialized parallel software and the study of the concentration of the deformations within the overlapping plate.

Mots-clefs

parallélisme, décomposition de domaine, automates cellulaires, systèmes dynamiques discrets, tectonique.

Key-words

parallelism, domain decomposition, cellular automata, discrete dynamical systems, plate tectonics.

Convention d'écriture

Les mots suivis d'un * sont placés dans l'index qui est situé en fin de mémoire, après la bibliographie.

Table des matières

Introduction générale

Le sujet et le contexte de l'étude

Cette thèse est une contribution à la modélisation du phénomène de subduction-érosion en tectonique des plaques. Cette modélisation par réseaux d'automates cellulaires a été entièrement amorcée le 1^er^ décembre 1995, date du début de financement de mon doctorat ; aucun travail préalable n'avait, jusqu'alors, été entrepris sur ce sujet précis. Elle est le fruit d'une collaboration entre le LGTE (Laboratoire de Géodynamique Tectonique et Environnement, CNRS - ESA 7073 et UFR 928 de l'Université Paris VI) et le LIP6 (Laboratoire d'Informatique de Paris VI, CNRS - UMR 7606 et UFR 922 de l'Université Paris VI).

L'idée générale de ce mémoire est de proposer et d'étudier successivement deux modèles discrets de ce phénomène géotectonique, puis de présenter les simulations informatiques correspondantes. Dans ce chapitre introductif, nous cherchons à justifier, à l'aide de considérations très générales, l'intérêt d'établir une simulation numérique et d'en envisager une optimisation parallèle. Nous terminons par une rapide présentation de l'organisation de ce mémoire.

De l'intérêt d'une modélisation*

Une modélisation, peut être soit une représentation mathématique ou logique synthétique, soit une représentation à échelle réduite ou dans des conditions limitées (modèle en bac à sable, modèle en soufflerie...), d'un système physique réel dans un contexte et une problématique donnés. Un modèle est quasiment toujours perfectible et doit-être validé par la réalité telle que nous la percevons.

L'intérêt d'une modélisation tient au fait qu'elle constitue une perception simplifiée de la réalité. Elle est de ce fait même plus facile à analyser, à conceptualiser ou à mettre en œuvre dans un processus simulatoire.

En géologie, la modélisation est souvent le seul moyen permettant de comprendre de façon quantitative le mécanisme d'un phénomène géodynamique et de prédire son évolution au cours du temps géologique. L'observation directe du globe terrestre actuel ne livre en effet qu'une image instantanée et fragmentaire de son évolution passée.

De l'intérêt d'une simulation* numérique

Une simulation numérique est une mise en œuvre (bien souvent informatique) du modèle. Il faut bien distinguer le temps de simulation (temps de calcul utilisé pour simuler un phénomène réel) du temps-réel. Une simulation numérique a plusieurs intérêts :

- il est possible de modifier facilement les paramètres de la simulation et d'observer ainsi leur influence sur le comportement global du système,
- il est possible d'avoir connaissance à tout moment de l'état du système (sans que cette mesure perturbe le comportement du système),
- une même expérience peut-être répétée autant de fois que nécessaire à peu de frais,
- une même expérience peut-être répétée autant de fois que nécessaire avec des conditions initiales strictement identiques.

Une fois l'investissement initial de modélisation du système et d'implémentation informatique effectué, la simulation numérique s'avère être un moyen d'investigation puissant et facile à mettre en œuvre.

De l'intérêt d'une modélisation discrète

Nous nous sommes placés résolument dans le cadre de la dynamique des systèmes. Cette discipline, développée au cours des années soixante, se propose de résoudre des problèmes liés à l'introduction de la dimension temporelle dans les études de comportement des systèmes complexes. Comme nous le verrons au cours de ce mémoire, nous sommes nous-mêmes confrontés à plusieurs échelles de temps.

Dans le contexte de la dynamique des systèmes, deux types d'approches bien distinctes se dégagent : les modèles sont ainsi dits continus ou discrets. Dans l'approche continue (encore appelée classique, avec des systèmes différentiels et des variables d'états), le milieu d'étude et le temps considérés sont tenus pour continus. Les éléments du système ne sont pas pris en compte en tant qu'individualités (ni dans leurs situations, ni dans leurs relations spatiales réelles). Seul le comportement du milieu dans son ensemble est réellement influent.

Dans le cas des modèles discrets, on part du principe que des comportements complexes peuvent être le résultat de la conjonction d'un certain nombre (fini) de règles élémentaires à portée locale. Ces modèles sont dits à base individuelle (individual-based models*) et sont tels que : "les interactions entre composantes élémentaires du système sont jugées déterminantes pour le comportement d'ensemble du système" [LANGLOIS and PHILIPPS, 1997]. Les règles locales y jouent un rôle initiateur essentiel dans l'émergence de comportements et de structures globales.

Comme le remarque [MÜLLER, 1996], l'approche continue "ne s'applique pas à tous les cas d'étude : [il est] difficile en effet d'observer des phénomènes comme la ségrégation [1], où les grains de petites tailles se séparent de ceux de grandes tailles, puisque les grains ne sont pas représentés en tant que tels".

Il existe à l'heure actuelle, un certain nombre de modélisations du phénomène de subduction. Certaines sont analogiques et se pratiquent en "boîte à sable" [2], d'autres sont plus analytiques et plus "globales" (méthodes classiques utilisant les équations aux dérivées partielles et la méthode des éléments finis. . .). Pour un certain nombre de raisons que nous exposerons plus tard au cours de ce mémoire, et parce que nous avons souhaité prendre en compte l'effet de l'érosion tectonique, nous avons choisi de modéliser le phénomène géotectonique étudié par un système numérique dynamique discret.

De l'intérêt des réseaux d'automates cellulaires et d'une approche de conception ascendante

Ayant opté pour les systèmes dynamiques discrets, nous avons considéré, pour des raisons que nous développerons lorsque nous passerons aux étapes de modélisation elles-mêmes, que l'approche par réseau d'automates cellulaires est la plus adaptée pour permettre de modéliser le phénomène de subduction-érosion. Pour ce qui est de la conception des réseaux d'automates cellulaires eux-mêmes, nous avons eu une démarche ascendante. Ainsi, nous sommes partis de l'idée que nous nous faisons des règles locales d'interaction mutuelle entre les composants élémentaires du milieu d'étude et nous en avons déduit les règles de fonctionnement du réseau d'automates cellulaires [3].

1. Nous serons, nous, confrontés à de l'érosion tectonique sous-crustale. . . mais le problème reste entier !

2. Il s'agit d'un dispositif de modélisation expérimentale que nous décrirons dans le chapitre sur les modèles existants.

3. Pour information : l'approche descendante correspond au fait de partir des lois physiques ou équations du modèle, pour retrouver, par "dérivation", les règles de transition locale du réseau d'automates cellulaires. Dans notre cas, cette approche aurait sûrement été très satisfaisante d'un point de vue théorique, mais nous n'avions aucune équation à notre disposition.

De la nécessité d'une approche "calculs distribués"

Il a souvent été constaté, lors du passage à la simulation informatique, qu'une approche discrète est plus coûteuse en temps de calcul (a priori) qu'une approche analytique. Jusqu'à un passé récent d'ailleurs, la capacité des ordinateurs constituait une véritable limite pour la mise en œuvre de simulations discrètes, du fait de temps de calcul prohibitifs.

C'est pour cette raison et aussi pour la raison que les réseaux d'automates cellulaires constituent un modèle canonique du calcul parallèle à fine granularité ([Delorme and Mazoyer, 1998] écrit ainsi à ce propos : "they are a natural model of parallelism"), que nous avons envisagé de mettre en œuvre une optimisation parallèle de notre simulation et de la porter sur un super-ordinateur massivement parallèle comme le Cray t3e. Nous constaterons d'ailleurs qu'il nous faudra développer notre propre plate-forme logicielle. En effet, les divers simulateurs que nous avons pu tester sont soit trop peu performants (souvent du fait de leur caractère conversationnel), soit inadaptés aux contraintes graphiques que nous nous sommes données, soit enfin trop pauvres en fonctionnalités lors de l'écriture même des fonctions de transition locale.

Plan de la thèse

Pour des raisons de lisibilité, nous avons délibérément choisi de fractionner l'ensemble de ce mémoire en un grand nombre de chapitres. En effet, du fait que cette étude se trouve à l'interface de plusieurs disciplines relativement peu habituées à cohabiter, nous espérons que ce découpage très prononcé facilitera le travail du futur lecteur, quelle que soit sa spécialité de rattachement.

Ce document est structuré en quatre parties. Après avoir présenté la problématique de l'étude et évalué ses enjeux, nous abordons les aspects généraux de nos deux simulations (théories, modèles, outils et principes de fonctionnement), avant de passer aux expérimentations elles-mêmes et à leurs résultats. Nous terminons ce mémoire par un bilan du travail effectué et de ses acquis puis par une présentation des perspectives de développements futurs.

Partie I : problématique et état de l'art

Cette partie a pour objectif de préciser le contexte géotectonique de la thèse en donnant les définitions, concepts et hypothèses que nous utilisons tout au long de ce document. Elle précise aussi les raisons qui sont à l'origine de ce travail et répertorie l'ensemble des restrictions que nous avons dû nous imposer pour mettre en œuvre nos deux modèles.

Elle est découpée en trois chapitres : le premier présente brièvement la tectonique des plaques, le second précise l'existant en matière de modélisation et les raisons et motivations qui amènent à développer nos propres modèles, le troisième, enfin, fait l'inventaire des diverses restrictions de modélisation et précise quelques ordres de grandeur.

Partie II : les modèles dynamiques discrets

Le principe d'une modélisation par système dynamique discret ayant été arrêté dans la partie précédente, nous présentons ici, en trois chapitres, le corps même de notre travail, en l'occurence : les modèles. Le premier chapitre précise et formalise les éléments théoriques nécessaires à la bonne compréhension des réseaux d'automates cellulaires ; il explique aussi les raisons qui poussent à choisir le formalisme "réseau d'automates cellulaires". Le second chapitre présente le modèle par réseau d'automates cellulaires uni-dimensionnel [Leduc, 1997, Leduc, 1998a], tandis que le troisième chapitre présente le modèle par réseau d'automates cellulaires bi-dimensionnel [Leduc, 1998b, Leduc, 1999]. Nous concluons cette partie par un rapide comparatif des deux méthodes ou modèles que nous avons développés.

Partie III : les simulations et l'optimisation parallèle

Dans cette partie, nous envisageons la mise en œuvre informatique des deux modèles exposés dans la partie précédente et son optimisation parallèle. Après avoir exposé les aspects matériels et logiciels ayant trait à l'algorithmique parallèle et répartie, nous présentons le choix architectural logiciel et matériel propre, puis les simulations.

Le premier chapitre répertorie les principales architectures de machines parallèles et présente les outils logiciels que nous serons amené à utiliser par la suite. Notre simulation et son optimisation parallèle ainsi que le portage de notre application sur le Cray t3e sont présentés, pour le cas uni-dimensionnel dans le second chapitre et pour le cas bi-dimensionnel dans le troisième chapitre. Nous concluons cette partie par un comparatif des deux bibliothèques d'échanges de messages que nous avons utilisées et montrons l'intérêt d'utiliser une bibliothèque d'échanges de messages appropriée dans le cadre d'une décomposition de domaine régulière sur une architecture parallèle à mémoire distribuée.

Partie IV : évaluation des performances, visualisation et interprétations

Dans cette dernière partie enfin, nous nous attachons à mesurer nos résultats selon deux critères. Ainsi, nous abordons successivement l'évaluation des performances de nos deux simulations du point de vue du parallélisme, puis présentons leurs rendus-visuels respectifs avant de conclure.

Première partie

Problématique et état de l'art

Chapitre 1

Tectonique des plaques et subduction

La tectonique des plaques* est une théorie bien établie depuis une trentaine d'années, selon laquelle la partie superficielle de l'enveloppe terrestre est constituée d'une mosaïque de calottes sphériques (les plaques) animées de mouvements relatifs dont l'origine est à rechercher dans la dynamique convective du manteau. Nous rappelons ici succinctement les notions utiles à notre étude.

1.1 Structure du globe terrestre

1.1.1 Stratification de l'enveloppe terrestre

La constitution du globe terrestre peut-être étudiée en analysant les informations fournies par la propagation des fronts d'ondes sismiques (artificielles ou naturelles) au sein de l'enveloppe terrestre. Mais elle peut aussi être envisagée au regard des informations gravimétriques. Nous allons développer successivement ces deux approches afin de présenter les différentes couches constitutives du globe terrestre ainsi que leurs mouvements verticaux ("radials") relatifs.

Comportement des ondes de compression et de cisaillement dans le globe

Un séisme provoque la libération quasi-instantanée d'une énergie élastique lentement accumulée. Partant du foyer (ou hypocentre), les vibrations atteignent la surface et se propagent de part en part à l'intérieur du globe, par réflexion, réfraction... Chaque particule minérale oscille autour de sa position d'équilibre initial, mais l'énergie associée se propage à grande vitesse. Les fronts d'ondes sismiques provoquent des déformations élastiques qui peuvent être caractérisées par le type de déplacement affectant les particules mises en mouvement. Plusieurs familles d'ondes sismiques se propageant à des célérités différentes sont ainsi recensées[1] :

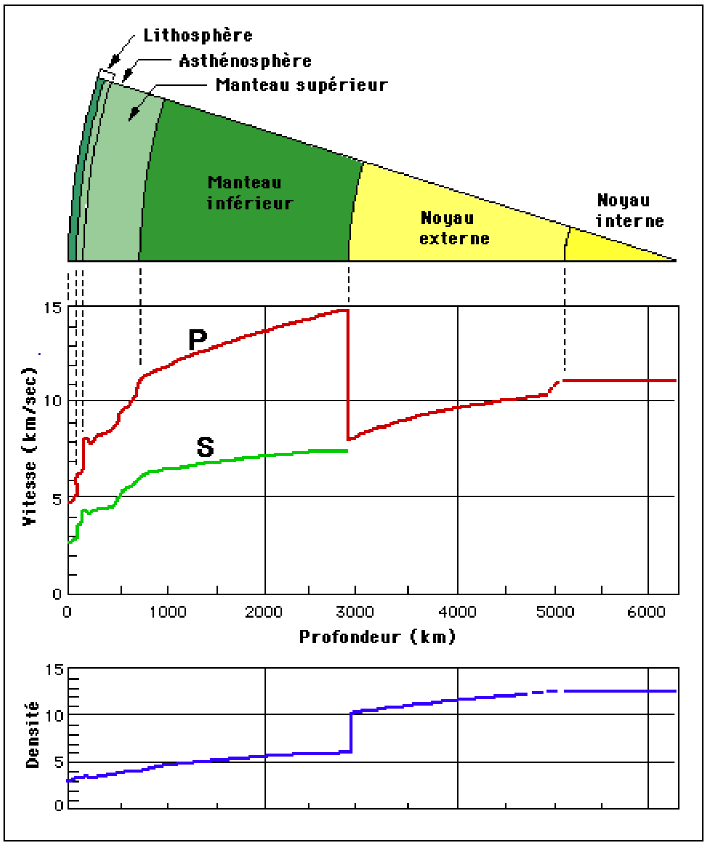

FIG. 1.1 – *Propagations des ondes P et S à l'intérieur du globe terrestre*

– les ondes longitudinales de compression P. Dans ce cas, les directions de déplacement des particules et de propagation de l'onde sont identiques. La déformation élastique se transmet de proche en proche, dans le sens du déplacement de l'onde, par compressions-dilatations successives ;

– les ondes transversales de cisaillement S. Cette fois, le déplacement des particules et la déformation s'effectuent dans un plan perpendiculaire à la propagation de l'onde. Ce type d'onde est susceptible d'être polarisé : verticalement (onde S_V) ou horizontalement (onde S_H) ; il est cantonné

1. On trouvera, par exemple, une explicitation des divers concepts qui sont exposés ici dans les ouvrages pédagogiques suivants : [BOILLOT, 1996], [DERCOURT and PAQUET, 1995], [DUBOIS and DIAMENT, 1997]...

aux milieux solides. Les ondes transversales S sont deux fois moins rapides que les ondes longitudinales P ;

- les ondes de Rayleigh. Ici, les particules décrivent une trajectoire elliptique à petit axe dans la direction de propagation de l'onde. Le sens de déplacement des particules est rétrograde par rapport à celui de l'onde. Ces ondes se propagent dans les interfaces.

Les ondes transversales S et longitudinales P (qui sont celles utilisées en sismologie) se propagent à l'intérieur du globe terrestre d'une façon qui dépend des propriétés physiques des divers milieux traversés. Leur vitesse de propagation [...] s'accroît avec la densité des terrains, et à densité égale, diminue au contraire si la rigidité (la viscosité) de ces terrains diminue. C'est par l'analyse des temps de propagation des ondes sismiques que l'on obtient des informations sur la nature interne propre du globe terrestre.

Structure sismologique de la terre

Ces analyses permettent de distinguer trois principaux milieux sismiques : l'écorce* (ou croûte*), le manteau* et le noyau* [CONDIE, 1997]. A l'interface entre l'écorce (l'enveloppe extérieure) et le manteau, il existe une zone appelée discontinuité de Mohorovicic* ou Moho* (voir fig. 1.2 et fig. 1.3). Cette zone, qui ceinture notre globe terrestre, se situe à une profondeur allant de quelques kilomètres sous l'hydrosphère*[2] à presque soixante-dix kilomètres sous les chaînes de montagnes. A l'interface entre le manteau et le noyau central se trouve une nouvelle zone de discontinuité sismique appelée discontinuité de Gutenberg* et située à 2900 km de profondeur.

[CONDIE, 1997] introduit encore quatre autres discontinuités à 50-200 km, 410 km, 660 km et 5200 km de profondeur. Ceci le conduit à présenter les différentes couches du globe suivantes :

- l'écorce est directement placée sous l'hydrosphère ou l'atmosphère et surplombant le Moho. Son épaisseur peut varier de 3 km (sur certaines rides médio-océaniques) à 70 km environ (sous certaines chaînes de montagnes) ;

- le manteau supérieur : s'étend du Moho jusqu'à la discontinuité des 660 km. Cette région comprend la zone de transition qui va de 410 km de profondeur à 660 km de profondeur ;

- le manteau inférieur : s'étend de la discontinuité des 660 km jusqu'à la discontinuité de Gutenberg (à 2900 km de profondeur). Cette région est encore appelée la mésosphère* ;

2. c'est-à-dire les océans ou les glaciers...

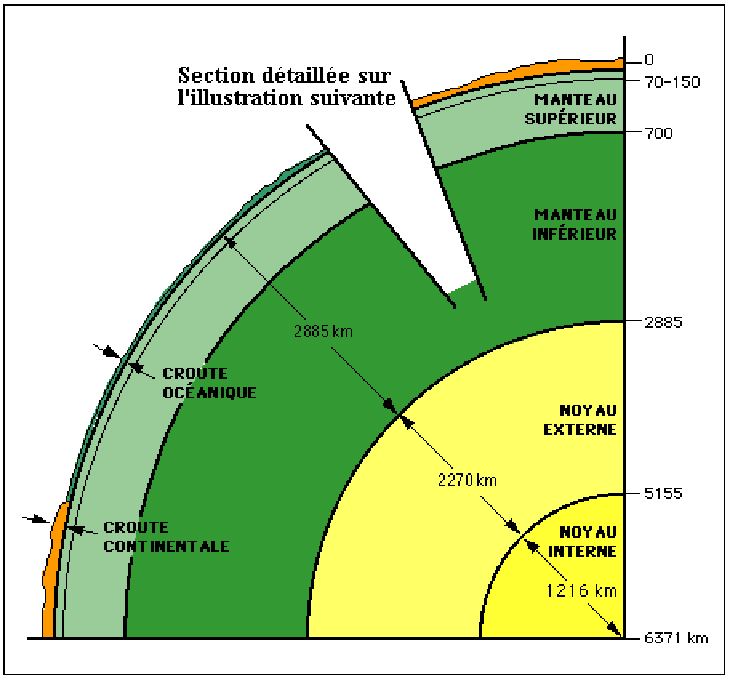

FIG. 1.2 – *Stratification du globe terrestre*

- le noyau externe : les ondes transversales S ne pouvant le traverser il doit être considéré comme un milieu liquide. Il s'étend de 2900 km de profondeur jusqu'à 5200 km de profondeur ;

- le noyau interne : s'étend de 5200 km de profondeur jusqu'au centre du globe terrestre.

Mais l'enveloppe terrestre peut encore être subdivisée de la façon suivante (pour sa partie supérieure du moins) :

- une lithosphère* comprenant l'écorce, le Moho et une petite partie du manteau supérieur (s'étendant jusqu'à une profondeur variant de 50 km jusqu'à 300 km). Cette enveloppe serait la plus rigide du globe (en grec *lithos* signifie pierre), sa viscosité est forte et les célérités des ondes y sont grandes ;

- une asthénosphère* s'étendant de la base de la lithosphère jusqu'à la discontinuité des 660 km. Cette région est d'une viscosité plus faible que la précédente (en grec *asthenes* signifie sans force), elle oppose une faible résistance aux contraintes et ne peut pas contrecarrer l'effet de la poussée d'Archimède. Ainsi, un objet géologique peut très bien traverser cette couche pour migrer vers la surface si sa densité est

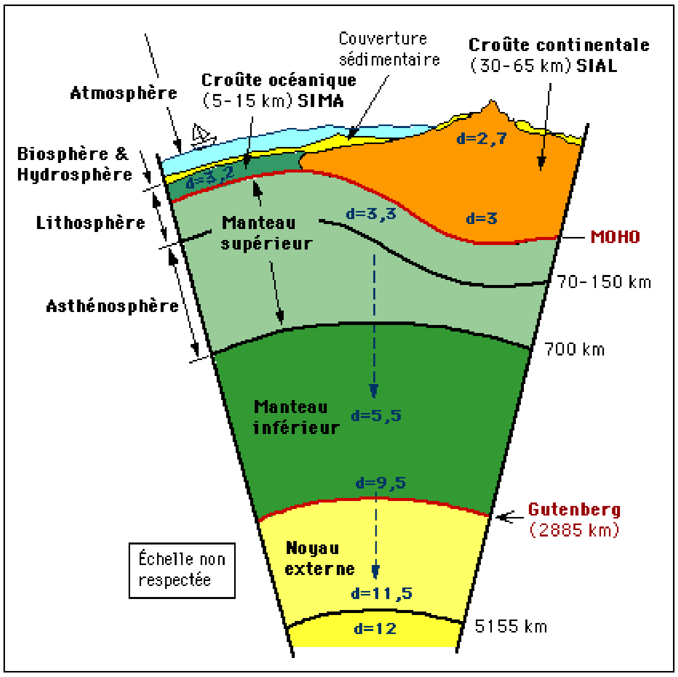

FIG. 1.3 – *Stratification du globe terrestre (suite)*

inférieure à celle du milieu ambiant ou au contraire "couler" vers les profondeurs si sa densité est plus forte.

Principe d'isostasie* - point de vue gravimétrique

La pesanteur est un invariant (il existe quelques anomalies de la gravité qui ne s'écartent pas de plus de 1/5000 de la valeur moyenne) quel que soit l'endroit où on la mesure à la surface du globe. Il est donc communément admis que la terre est formée de couches concentriques, ce qui implique que la pression imposée en un point quelconque de l'intérieur du globe est constante à profondeur constante.

En fait, ce modèle très simple de répartition des pressions ne s'applique qu'à la partie interne de la terre, où la faible résistance des terrains permet aux pressions de s'égaliser à profondeurs égales. Ainsi, pour reprendre un exemple extrait de [BOILLOT, 1996], si l'on se place à 10 km sous un continent d'altitude 0, la pression[3] est de 2,8 kbar (puisque la densité de la croûte continentale est de 2,8). Par contre, si l'on se place, à la même profondeur, mais sous 5 km d'eau et 5 km de croûte océanique (densité de

3. Nous avons délibérément choisi de ne pas utiliser les unités du système international pour des raisons de lisibilité d'abord, mais aussi pour respecter les usages en vigueur dans la discipline.

la croûte continentale : 2,9) elle n'est plus que de 1,95 kbar. Entre ces deux points pourtant situés à la même profondeur, la rigidité de la lithosphère interdit toute égalisation des pressions. Pour que la gravité soit la même quasiment en tout point du globe, c'est-à-dire, pour que les différences dans la répartition superficielle des masses soient compensées, on admet l'existence d'une surface de compensation* sous la lithosphère où les pressions s'égalisent. C'est le principe d'isostasie*. Cette correction isostatique s'effectue dans l'asthénosphère où, grâce à l'absence de rigidité du milieu, les pressions deviennent pratiquement hydrostatiques* (c'est-à-dire isotropes).

En fait, les lois de l'isostasie découlent du principe d'Archimède : la croûte "flotte" sur un manteau plus dense, la "poussée" du manteau sur la croûte est comparable à la poussée de l'eau sur un objet flottant.

Les mouvements verticaux de la lithosphère

Sachant que les reliefs montagneux s'érodent au cours des temps géologiques, on peut en déduire que la croûte s'amincit par "ablation superficielle". Pourtant, l'érosion d'une tranche de terrain n'implique pas pour autant l'abaissement du relief. En effet, éroder en surface une épaisseur de terrain de 1 km (de densité 2,8), implique, d'après le principe d'isostasie, de lui ajouter en profondeur une tranche de terrain mantellique (de densité 3,3) de $1\text{km} \times 2,8/3,3 \simeq 0,85$ km. Ainsi, une grande partie du relief érodé est reconstituée par un soulèvement régional et une migration du Moho vers la surface (comme le ferait la ligne de flottaison d'un navire en cours de déchargement). Cet effet de compensation permet de mettre en évidence l'existence de mouvements verticaux : nous pouvons presque parler d'un "effet bouchon de liège".

Par ailleurs, il est aussi possible d'envisager des mouvements verticaux de la lithosphère sous l'effet d'une variation de son état thermique. En effet, supposons un réchauffement régional lié à du volcanisme provoquant une diminution de la densité de la lithosphère de 0,01. Alors, l'application de l'hypothèse de l'équilibre isostatique local à une lithosphère de 120 km d'épaisseur subissant un tel réchauffement, la conduit à se soulever de 1,2 km. Inversement, il est possible d'envisager une subsidence* de la lithosphère, c'est-à-dire un enfoncement de sa surface, sous l'effet d'un refroidissement. C'est exactement ce qui se passe pour une lithosphère océanique[4] dont le "toit" plonge de 2,5 km de profondeur au niveau d'une ride médio-océanique, à plus de 5,5 km au niveau d'une fosse océanique sous l'effet du refroidissement de la croûte magmatique venue de la ride[5].

4. Cette appellation ainsi que celles qui suivent dans cette phrase seront présentées par la suite.

5. Croûte faite de basalte, gabbro ou serpentinite, qui s'est formée à la suite d'une fusion anhydre du manteau

1.1.2 La lithosphère* terrestre

Comme nous l'avons déjà écrit, la lithosphère est située directement sous l'atmosphère ou sous l'hydrosphère. Il ne faut pas, pour autant, la considérer comme un ensemble homogène. On distingue donc, une lithosphère océanique, placée majoritairement sous l'hydrosphère, d'une lithosphère continentale, surmontée par l'atmosphère.

Les différents types de lithosphère

La différence majeure entre la lithosphère océanique et la lithosphère continentale a trait à leurs densités moyennes respectives. En effet, la lithosphère océanique a une densité de 3,3 qui est légèrement supérieure à celle de l'asthénosphère dont la valeur est de 3,25 [6]. Elle est donc destinée à disparaître en subduction "peu de temps" après sa formation. La lithosphère continentale, elle, au contraire, est d'une densité de 3,2 un peu plus faible que celle de l'asthénosphère. Son destin est donc de "rester" à la surface du globe au fur et à mesure de sa formation. On trouvera la valeur de ces paramètres physiques dans [Ahrens, 1995].

Cartographie des plaques lithosphériques

[Condie, 1997] distingue 7 plaques majeures (tailles de l'ordre de 10^8km^2) : l'Antarctique, l'Eurasie, l'Amérique du Nord, l'Amérique du Sud, le Pacifique, l'Afrique et l'Australie. Il recense aussi quelques plaques de tailles intermédiaires (de l'ordre de $10^6 - 10^7$km^2) comme : les Philippines, l'Arabie, Nazca, les Cocos, les Caraïbes et la mer de Scotia. Enfin, il dénombre encore plus de 20 plaques d'une superficie supérieure à 10^5km^2.

1.2 Dynamique des plaques

Comme nous l'avons déjà écrit, les plaques lithosphériques ne sont pas immobiles. Elles sont susceptibles d'avoir des mouvements verticaux en réponse à des variations de l'épaisseur crustale, de leur état thermique... Et ce n'est pas tout, elles ont aussi des mouvements "horizontaux". C'est d'ailleurs à partir de cette constatation et de ses implications morphologiques que, dès le début du siècle, Wegener proposa une théorie de dérive continentale.

Ainsi, chaque plaque est assimilée à un ensemble rigide dont le mouvement à la surface du globe peut-être décrit comme une rotation autour d'un axe passant par le centre de la sphère [Le Pichon, 1968, Le Pichon et al., 1973]. Commençons par évaluer les forces ayant un rôle moteur dans le déplacement des plaques.

6. Ce qui s'explique par le fait que les matériaux de la lithosphère océanique et de l'asthénosphère sont sensiblement les mêmes ; mais que ceux de la lithosphère sont plus froids et donc plus denses que ceux de l'asthénosphère.

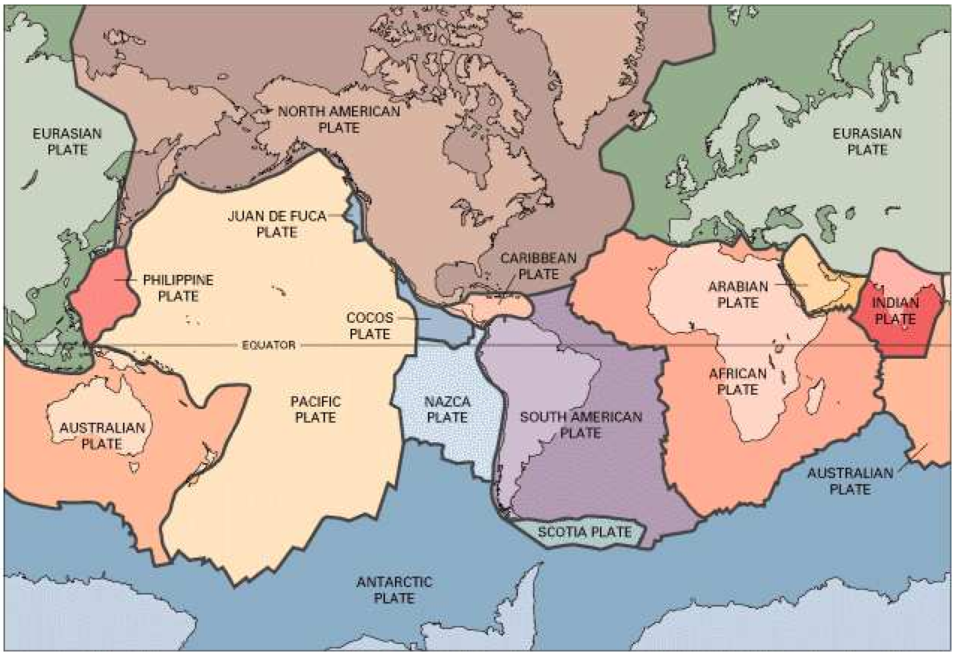

FIG. 1.4 – *Les principales plaques lithosphériques*

1.2.1 La convection mantellique*

Dans le modèle de Rayleigh-Bernard, la convection au sein d'un fluide dépend du nombre de Rayleigh* (*Ra*, sans dimension). Ainsi, pour qu'un liquide simple et homogène chauffé à sa base entre en état de convection, il faut que son nombre de Rayleigh dépasse la valeur 2000 [CONDIE, 1997]. Dans le manteau terrestre, la valeur de ce nombre serait de l'ordre de 5.10^6 [LAFAURIE, 1995], ce qui nous place dans le cas d'une convection turbulente irrégulière[7]. Les axes de convection de l'asthénosphère sont horizontaux, mais deux hypothèses s'affrontent en ce qui concerne l'organisation de la convection mantellique : elle pourrait se produire dans un système à une couche ou à deux couches.

D'après [CONDIE, 1997], on peut répartir les forces à l'origine du mouvement horizontal des plaques de la façon suivante : 95% des efforts pro-

7. Il importe ici de se rappeler que le gradient de température entre les profondeurs et la surface de l'écorce terrestre est élevé. On estime qu'il existe un isotherme de 1300°C situé vers 120 km de profondeur à la base de la lithosphère. A la base de la mésosphère, au niveau de la discontinuité de Gutenberg, la température pourrait varier de 2500 à 5000°C. Il semble que la source d'énergie thermique provienne de la désintégration d'éléments radioactifs dans le manteau et le noyau. Par contre, entre la base de la lithosphère et la surface du globe, où la viscosité est élevée, la convection n'est plus possible et la chaleur se transmet par conduction. Ce transfert d'énergie est très lent et dépend de la conductivité thermique des roches.

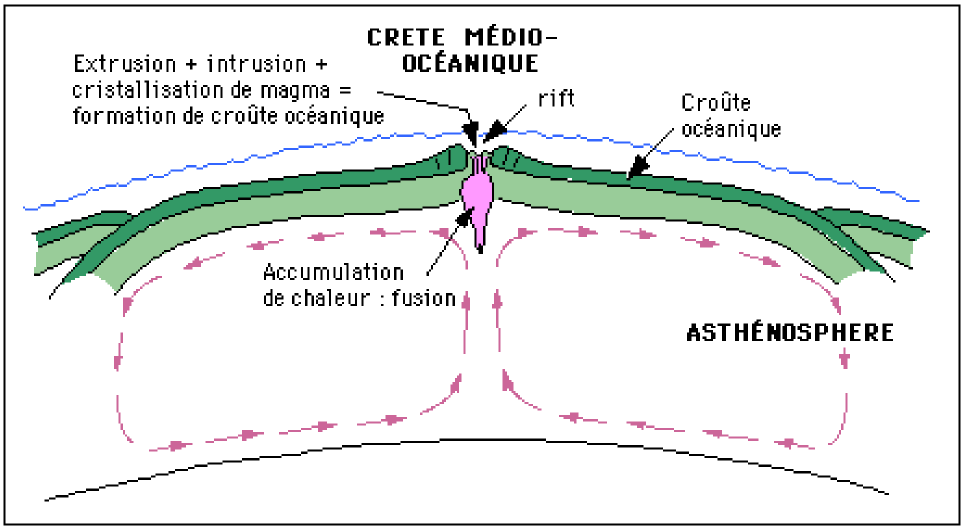

FIG. 1.5 – *Convection mantellique : système à une couche*

viendraient d'un effet de traction du front de la plaque par les profondeurs et les 5% restant se partageraient entre une poussée au niveau de la ride médio-océanique (à l'endroit même où la plaque se forme) et un effet "tapis roulant" à la base de la plaque (par frottement mécanique).

1.2.2 Conséquences de mouvements horizontaux de la lithosphère

Les mouvements lithosphériques et asthénosphériques sont couplés. Ainsi, lorsque deux cellules de convection asthénosphériques adjacentes ont des sens de rotation contraires mais un même axe de rotation horizontal, les plaques lithosphériques correspondantes peuvent converger ou diverger. Si, au contraire, les axes des cellules de convection sont transversaux à l'interface, les plaques lithosphériques coulissent l'une contre l'autre. Dans chacun des cas, les déplacements sont de l'ordre du centimètre ou de la dizaine de centimètres par an[8].

Comme le constate [JUTEAU, 1993], les plaques se forment au niveau des rides médio-océaniques* par montée continue de magma basaltique. La lithosphère ainsi créée se refroidit et s'éloigne de la dorsale* (la ride), pour disparaître, en subduction, au niveau des fosses océaniques* dans les zones dites de subduction. Au fur et à mesure qu'elle s'enfonce dans l'asthénosphère, la plaque lithosphérique se réchauffe et "fond" partiellement donnant

8. Pour [CONDIE, 1997], l'Amérique du Nord se déplace à la vitesse de 1,5-2 cm/an, l'Eurasie à 2,4-2,7 cm/an, l'Afrique à 2,9-3,2 cm/an, l'Australie à 6-7,5 cm/an et le Pacifique à 5-7 cm/an...

naissance à des magmas qui remonteront à leur tour et viendront s'accréter à la marge*[9] continentale, participant ainsi à sa croissance.

1.2.3 Les types d'interface

D'après [Kearey and Vine, 1996], on peut assimiler une plaque lithosphérique à un ensemble relativement rigide et considérer que, lorsque un effort est appliqué à l'une de ses extrémités, celui-ci est transmis de bout en bout sans déformation intraplaque majeure. Les déformations n'ont lieu que sur les bords (les marges). [Condie, 1997] recense trois types majeurs à l'interface des marges :

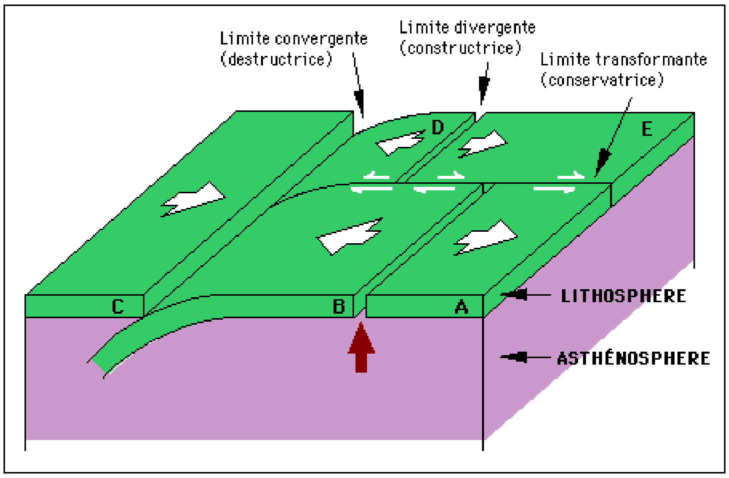

Fig. 1.6 – *Les trois principaux types de marges*

- les marges divergentes*. L'écartement de deux marges peut être compensé soit par un amincissement de la lithosphère (lorsque ces plaques portent une croûte continentale) avec création d'un rift continental*, soit par la création, à la verticale de la dorsale, de nouveau matériel lithosphérique aux dépens de l'asthénosphère (lorsque ces plaques sont en partie de nature océanique) avec un rift océanique*. On parle alors de marge passive*. Il s'agit d'un processus additif* d'expansion de marges, avec création de surface terrestre ;
- les marges convergentes*. Elles sont caractérisées par une fosse océanique* qui marque la frontière entre la plaque chevauchante et la plaque en subduction lorsqu'il y a convergence d'une marge océanique et d'une marge continentale. Dans le cas d'une convergence continent-continent, il se produit un phénomène de collision continentale*. On parle alors de

9. L'extrémité d'une plaque lithosphérique est appelée marge.

marge active*. Il s'agit ici d'un processus soustractif* avec destruction de surface terrestre ;

- les failles transformantes*. Elles se forment à l'interface entre deux plaques coulissantes[10]. On parle alors de marge transformante*.

1.3 Marges convergentes ou marges actives

Dans le cadre de nos modélisations, nous nous sommes plus particulièrement intéressés au cas des marges convergentes, avec passage d'une lithosphère océanique sous une lithosphère continentale. En fait, nous nous sommes même fortement inspirés du cas de la fosse d'Amérique centrale (localisée dans le Pacifique, au large de Mexico à l'interface entre la plaque des Cocos et la plaque d'Amérique du Nord) qui est l'une des zones de subduction les mieux étudiées [Bourgois, 1993]. Commençons d'abord par décrire le phénomène de convergence plus précisément.

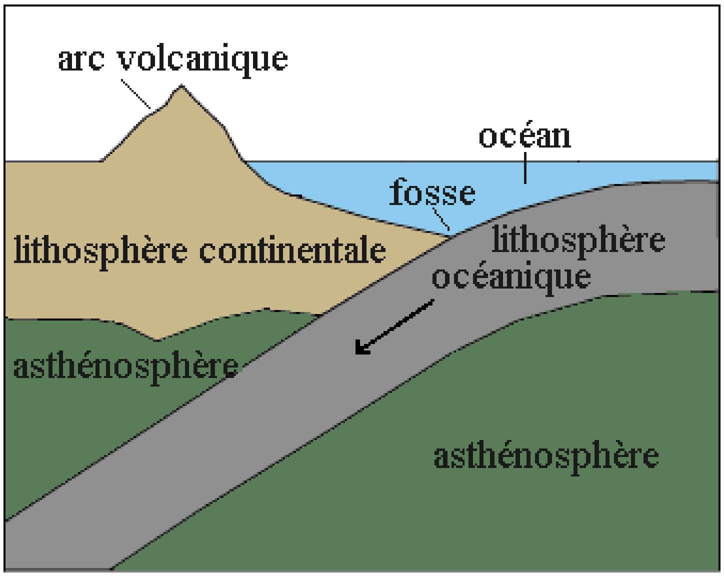

Fig. 1.7 – *Subduction d'une lithosphère océanique sous une lithosphère continentale*

10. C'est le cas de la faille de San Andreas en Californie, par exemple...

1.3.1 Bilan des forces tectoniques en présence

Le plongeon d'une plaque océanique sous une plaque continentale, implique d'abord, d'après [Forsyth and Uyeda, 1975, Carlson and Uyeda, 1983], l'ensemble des efforts suivant à prendre en compte :

- un couplage de la base de la lithosphère avec l'asthénosphère,
- un entraînement gravitaire de la marge froide qui subduit,
- une résistance du manteau à la pénétration de la marge en subduction,
- des transformations chimiques (avec gain de masse) accompagnant la déshydratation de la croûte subduite vers 50-70 km et vers 400 km de profondeur.

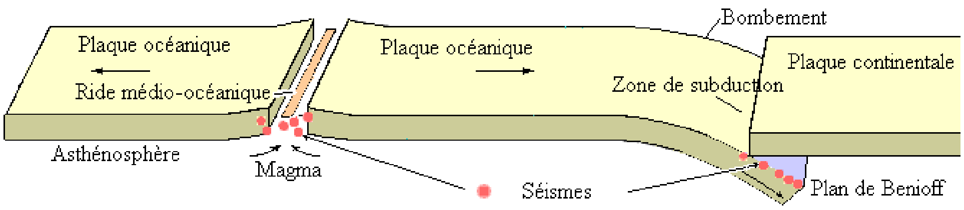

Fig. 1.8 – *Mécanisme de subduction*

1.3.2 Signature morphologique de la subduction

Du fait de l'effort gravitaire exercé par la plaque chevauchante sur la plaque subduite et du fait de la rigidité flexurale de la lithosphère superficielle subduite, il y a création d'une dépression (la fosse océanique*) en bordure du lieu d'application de la charge (à l'avant de la marge continentale chevauchante) et d'un "bombement" de la marge subduite un peu en retrait, à l'avant de la fosse, du côté océanique.

Comme la surface de la marge subduite augmente sous cet effort de flexion, il se produit un phénomène d'extension*. La lithosphère superficielle fragile se brise alors localement, des failles normales et de petits fossés d'effondrement apparaissent et tout le bombement est ainsi le siège d'une activité sismique superficielle faible...

Par ailleurs, dans le pourtour immédiat d'une zone de subduction, on observe aussi les phénomènes suivant : surpressions de fluides le long du plan de subduction (au niveau de ce que certains appellent le chenal de subduction*, dans la zone de subduction proprement dite) [11], production de séismes par frottement ou relaxation [12], création d'un arc volcanique* [13]...

1.3.3 Subduction érosion et subduction accrétion

En principe, le destin de la croûte continentale est donc de s'accroître par accrétion de croûte, tandis que celui de la lithosphère océanique est de disparaître rapidement (à l'échelle géologique) de la surface du globe terrestre, en s'engloutissant dans l'asthénosphère sous les marges et arcs insulaires actifs [von Huene and Scholl, 1991]. Ainsi, il n'y a pas de lithosphère océanique de plus de 180 Ma [14] à la surface du globe. La réalité est en fait plus complexe, et plusieurs processus concurrents contribuent à l'accrétion continentale ou la contrarient.

[Scholl et al., 1980] regroupent l'ensemble des marges en subduction [15] en trois régimes bien distincts : d'accrétion, intermédiaire et érosif. A chacun de ces régimes est associé une géométrie de marge particulière.

Un processus additif ou accrétif : régime en compression (type I)

Quand une plaque océanique s'enfonce dans une zone de subduction, les sédiments qui la recouvrent [16] peuvent être "avalés" par elle ou retenus en surface dans l'interface entre les deux plaques.

Dans le premier cas, la couverture sédimentaire qui reste adhérente à la croûte océanique subduite et la suit dans la subduction s'appelle le tégument*.

Dans le deuxième cas, la plaque continentale chevauchante a un effet de rabot et il se constitue à son front une imbrication d'écailles tectoniques constituées du matériel sédimentaire déposé sur le plancher océanique basaltique [Bourgois, 1993]. La subduction de la plaque inférieure et de sa couverture sédimentaire s'accompagne d'une déformation en compression* qui se traduit par l'accumulation d'un bourrelet sédimentaire au front de

11. A ce propos, le lecteur pourra lire avec profit la référence [Bourgois, 1993], où il est question des surpressions au niveau du forage du site 567 lors du Leg 84...

12. Car la couche fragile superficielle de la plaque plongeante est soumise à d'intenses contraintes. Les foyers de ces séismes sont enfermés dans un volume parallélépipédique très aplati, nommé plan de Wadati-Benioff*.

13. Un arc volcanique est un ensemble de volcans disposés à une distance d'environ 150 km de la fosse le long d'un arc.

14. Comme nous l'avons déjà signalé, nous n'utilisons pas les unités du système international pour respecter les usages en vigueur dans la discipline.

15. Il y a 40000 km de frontières convergentes à la surface du globe et 90% des zones de subduction se trouvent dans l'océan Pacifique.

16. Les sédiments qui recouvrent la lithosphère subduite peuvent être océaniques, parce que déposés depuis la naissance de la plaque, ou terrigènes, parce que transportés depuis le continent par des fleuves et des courants marins.

la marge supérieure appelé le prisme d'accrétion* [17]. Ce sous-charriage peut engendrer des prismes d'accrétion de quelques dizaines de kilomètres, voire même de plusieurs centaines de kilomètres de largeur sur plusieurs milliers de kilomètres de longueur.

Un processus soustractif ou érosif : régime en extension (type II)

Toutes les marges convergentes ne sont cependant pas accrétives (21000 km des 40000 km linéaires de marges convergentes à la surface terrestre sont dites de type II, c'est-à-dire en extension, [Bourgois, 1993, Aubouin et al., 1984]). En effet, ces références expliquent qu'une abrasion de la base de la plaque supérieure associée à la subduction de la plaque inférieure conduit à un affaissement (une subsidence*) et un glissement de la plaque supérieure. Ce glissement provoque une migration, un retrait plus précisément, du front de la marge chevauchante et de l'axe central de la fosse océanique en direction du continent. D'après [Bourgois, 1993], la fosse du Pérou a migré, comme chaque point de la marge, de 100 km en direction du continent en 5 Ma environ.

Marges où l'érosion et l'accrétion sont constatées : régimes intermédiaires

Enfin, [von Huene and Scholl, 1991] propose un troisième type de marges en régime intermédiaire. C'est-à-dire des marges où accrétion et érosion tectonique sont constatées, ou dont le bilan de matière est indéterminé car les volumes accrétés et/ou érodés sont inconnus.

Les indicateurs de l'érosion

[von Huene and Scholl, 1991] fait une évaluation des critères de reconnaissance de l'érosion tectonique :

- retrait simultané de l'arc et de la fosse (indicateur potentiel),
- proximité du socle continental avec la fosse (indicateur potentiel),
- augmentation de la pente topographique de la marge supérieure en bordure de la fosse, au niveau du prisme (indicateur potentiel),
- troncature des niveaux à la base du prisme (indicateur potentiel),
- augmentation de la couche sédimentaire sous-plaquée en fonction de l'enfouissement depuis la fosse (indicateur relativement significatif),

17. C'est par exemple grâce aux énormes apports sédimentaires de l'Amazone qu'a pu se constituer le prisme d'accrétion de la Barbade qui ceinture les Antilles à l'est dans l'océan Atlantique. Ce prisme s'est tellement épaissi qu'il a d'ailleurs fini par émerger...

– forte subsidence de la marge chevauchante en l'espace de quelques millions d'années (indicateur significatif).

1.3.4 Moteurs de l'érosion et de l'accrétion

A ce niveau, plusieurs hypothèses s'affrontent. Nous en présentons deux, avant d'aborder celle que nous retiendrons, qui lie l'angle d'ouverture de la zone de subduction au type de régime (en extension ou en compression) [CLOOS and SHREVE, 1996].

[SCHNÜRLE, 1994] établit ainsi "une bonne corrélation entre vitesse importante et régime érosif/vitesse faible et régime d'accrétion". Pour [LALLEMAND, 1992], une forte pente du décollement* [18], caractéristique d'un fort ploiement de la plaque en subduction, entraîne un enfouissement plus rapide, d'où une augmentation de la surpression de fluides à la base de la plaque supérieure et, conséquemment, une augmentation de l'hydrofracturation et donc de l'érosion sous-crustale. Mais, comme [SCHNÜRLE, 1994], [LALLEMAND, 1992] estime également qu'une forte vitesse de convergence entraîne aussi une augmentation de l'hydrofracturation de la base de la plaque chevauchante et donc de l'érosion tectonique [von HUENE and CULLOTTA, 1989, von HUENE and LALLEMAND, 1990].

Le cadre d'hypothèses dans lequel nous nous plaçons est différent. Nous partons du principe que la fermeture de l'angle du "chenal de subduction" (c'est-à-dire la diminution du pendage de la surface de la plaque subduite, en rapport avec l'augmentation de sa vitesse de plongeon) doit entraîner un effet d'accrétion alors qu'une ouverture de cet angle (c'est-à-dire l'augmentation du pendage de la plaque subduite, en rapport avec sa perte de vitesse et son gain de poids) produit un effet d'érosion [CLOOS and SHREVE, 1996].

En effet, une "fermeture" du chenal de subduction peut-être la cause, par effet d'obturation, d'un refoulement de la matière subduite hors du chenal de subduction avec accrétion à la base et au front de la marge chevauchante, tandis qu'une "ouverture" de ce même chenal, du fait même de la libération de la zone de chenal, entraîne une augmentation de volume interplaque et d'érosion sous-crustale (pénétration dans le chenal d'une plus grande quantité de matériel sédimentaire à forte teneur aqueuse, d'où une plus grande déshydratation et un effet d'hydrofracturation majoré) [CLOOS and SHREVE, 1996].

Influence de la subduction d'une aspérité sur la variation de régime

Comme on peut le remarquer sur la fig. 1.9, l'ouverture du chenal de subduction (par augmentation du pendage de la plaque en subduction) est

18. Le décollement, c'est la zone de quelques mètres d'épaisseur délimitant deux unités structurales dont les régimes de contraintes sont différents (dans notre cas la limite de plaque) [SCHNÜRLE, 1994].

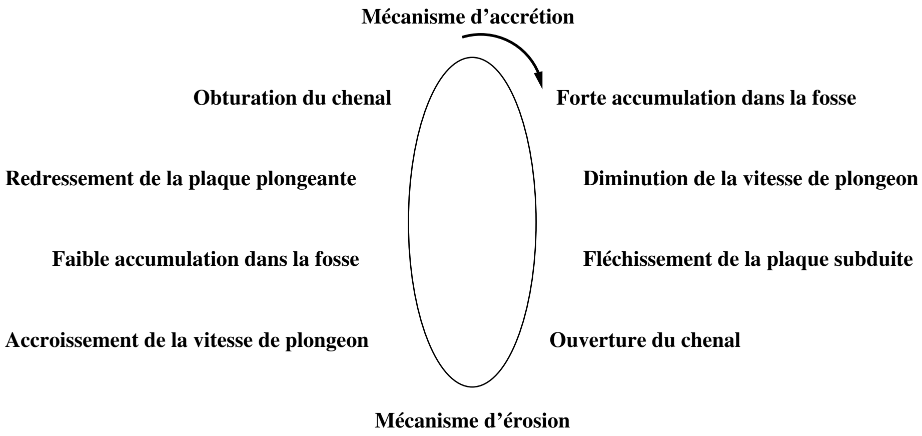

FIG. 1.9 – *Diagramme des mécanismes en jeu lors d'une subduction*

associée au ralentissement de la vitesse de plongeon de la plaque inférieure. Ce mécanisme de ralentissement pourrait permettre à une zone en régime de subduction-accrétion de passer à un régime de subduction-érosion. Une des causes du ralentissement du plongeon de la plaque en subduction pourrait être la subduction d'aspérité.

Chapitre 2

Les modèles existants : de la nécessité d'une autre approche

Avant le commencement de nos travaux, la simulation du phénomène de subduction était principalement abordée sous l'angle de méthodes analogiques ou de méthodes numériques. Au cours de ce chapitre, nous présentons succinctement ces deux approches avant d'en étudier les avantages et inconvénients respectifs et d'argumenter la nécessité de développer une démarche radicalement différente.

2.1 L'existant

Dans sa préface, [SHEMENDA, 1994] écrit :

> « *Of the two modeling methods, mathematical (numerical) and physical (experimental), I would prefer dealing with a computer, leaving the "dirty" experimental work to those who enjoy working with real physical objects. [...] However, this method*[1] *encounters serious problems in constructing ever more complex and adequate models of geologic processes in spite of the great possibilities provided by modern supercomputers. [...] In many cases, physical modeling can in principle obviate these difficulties which makes this method more attractive and deserving of further development. The main problem is to put the experiments on a quantitative basis, i.e. to create models that satisfy the necessary requirements, the similarity criteria, in the first place.* »

1. Il s'agit bien sûr ici de l'approche "mathématique"...

Ainsi, lorsque l'on procède par modélisation expérimentale, c'est-à-dire, lorsque l'on cherche à reproduire un phénomène géotectonique à l'échelle réduite d'un laboratoire, il est nécessaire de satisfaire un certain nombre de conditions appelées "critères de similarité* physique". Commençons par présenter l'approche analogique.

2.1.1 Simulations analogiques : l'approche expérimentale

La modélisation analogique consiste à reproduire, dans une boite rectangulaire en verre transparent (ce qui permet une observation "continue et non destructive de l'évolution expérimentale" comme le remarque [Malavieille et al., 1993]) de quelques dizaines de centimètres, des événements géologiques correspondant à des portions du globe de plusieurs kilomètres et sur des périodes de plusieurs millions d'années. Une plaque de PVC recouverte d'une mince couche de sable[2] qui est sensée représenter le dépôt sédimentaire de la croûte océanique, est mise en mouvement. Un butoir rigide placé en surface sur l'extrémité finale de la plaque de PVC représente le socle continental (ou l'arc continental). L'évolution dynamique de la formation du prisme et de sa déformation interne peut-être observée au travers des parois latérales en verre.

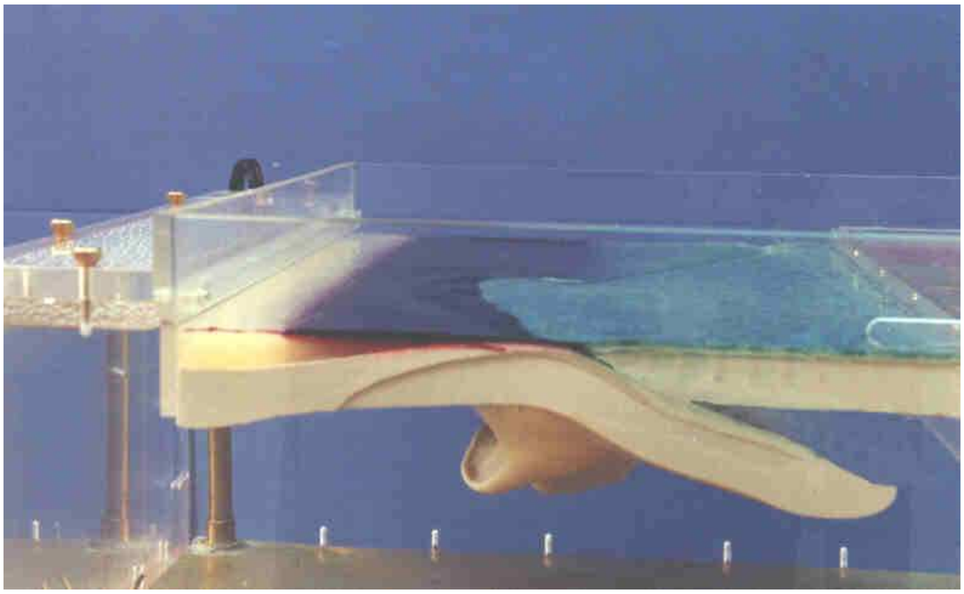

Fig. 2.1 – *Dispositif de modélisation expérimentale (extrait de [Shemenda, 1994])*

2. D'après [Schnürle, 1994], "le sable est un bon analogue du comportement fragile de la croûte supérieure"...

Comme le souligne [Shemenda, 1994], "the main difficulties in the physical modeling are associated with the creation of model materials and experimental conditions that satisfy the physical similarity conditions". En d'autres termes, il s'agit de déterminer les paramètres physiques des matériaux utilisés pour modéliser l'écorce terrestre, le manteau et la couverture sédimentaire de façon à ce qu'ils respectent, dans une proportion acceptable, la taille de l'objet naturel étudié ainsi que son comportement sur le pas de temps considéré. Pour ce faire on utilise, par exemple, des matériaux composites constitués d'alliages d'hydrocarbures solides, d'huile minérale, de différentes poudres finement broyées et de matériaux actifs en surface... Ils possèdent des propriétés élasto-visco-plastiques[3] complexes qui dépendent fortement de leur composition, de la température et des conditions de mise en œuvre du matériau. La variation de ces conditions permet d'obtenir les valeurs des paramètres mécaniques et thermo-physiques des matériaux nécessaires pour satisfaire les critères de similarité.

Ainsi, [Kincaid and Olson, 1987] précise, dans le cadre des procédures expérimentales, que le modèle a nécessité la mise en place de feuilles de saccharose fortement concentré, de 1 cm d'épaisseur, refroidies à $-15°$ C pour respecter des conditions de haute viscosité. Pour [Jolivet, 1995], "le sable de Fontainebleau sec est un très bon milieu pour faire apparaître des failles, et son comportement est celui d'un matériau fragile[3], à faible cohésion". Pour [Malavieille et al., 1993], le sable sec tamisé à 50 μm constitue un bon analogue des matériaux impliqués dans les déformations superficielles de l'écorce terrestre. [Lallemand et al., 1994] précise, lui, que "the sand is essentially cohesionless and its deformation is time-independent". [Jolivet, 1995] note encore que "les silicones ont un comportement qui varie fortement avec la température, la contrainte appliquée et le taux de déformation. En utilisant des vitesses raisonnables, son comportement est ductile[3], elle peut donc servir à modéliser la croûte inférieure".

Ainsi, dans une expérience analogique, on utilise des matériaux qui ont a priori un comportement proche de celui de la lithosphère (sable : cassant, silicone : ductile), et on leur applique des conditions aux limites variées. Une telle manipulation est validée par comparaison avec les observations et mesures géologiques et géophysiques effectuées sur le terrain. [Shemenda and

3. Un comportement plastique* de roche correspond à une roche dont la déformation en réponse à la contrainte n'est pas réversible et la relation entre contrainte et déformation n'est pas linéaire. Les matériaux plastiques ne restituent pas la déformation après relaxation de la contrainte. Par opposition, on définit un comportement élastique* de roche lorsque celle-ci se déforme instantanément quand elle est soumise à une contrainte, et dont la déformation est réversible. Dans ce cas, la relation entre déformation et contrainte est souvent linéaire. Les matériaux élastiques accumulent une déformation qu'ils restituent lorsque la contrainte est relâchée. On introduit ensuite la notion de comportement cassant* ou fragile* lorsque les roches ne subissent pas ou peu de déformation plastique avant la rupture. Dans le domaine ductile*, les roches subissent de grandes transformations plastiques.

GROCHOLSKY, 1992], en présentant un nouveau modèle "à quatre couches", mentionne plus d'une quinzaine de modèles analogiques. Les limites de ces modèles sont exposées dans le paragraphe 2.2 suivant.

2.1.2 Simulations analytiques et numériques "globales"

Elles ont le même but que la modélisation expérimentale mais cette fois les moyens sont différents : on crée un modèle mathématique en décrivant les processus tectoniques naturels par des équations. Les équations sont d'ordre élevé et, normalement, elles ne peuvent être résolues qu'à l'aide d'ordinateurs, en utilisant des méthodes numériques.

[DUBOIS, 1995], après avoir présenté le modèle de lithosphère élastique à bord libre de Hanks (1971), le modèle élasto-plastique de Turcotte (1978), le modèle visqueux de De Bremaecker (1977), le modèle visco-élastique de Liboutry (1974) ou Ida (1984) et le modèle de Melosh (1978) conclut que l'utilisation de techniques par éléments finis permet d'envisager des rhéologies* [4] plus complexes et de tenir compte de la variabilité de la lithosphère en fonction des conditions physiques existantes.

Ainsi, après avoir défini les contours de la ou des géométries étudiées, il faut mailler [5] l'ensemble du domaine à l'aide de polytopes jointifs partitionnant l'ensemble de l'objet virtuel. A chacun de ces éléments polyhédriques est associé une loi de comportement : élasticité linéaire, plasticité parfaite, visco-plasticité, élasticité non linéaire, affaiblissement et durcissement... A chacun de ces éléments on associe des forces extérieures appliquées à l'objet : forces de volume (le poids lié à la densité par exemple), forces de contact sur les frontières et forces ponctuelles [SCHNÜRLE, 1994]. Enfin, il faut définir les conditions de déplacement aux frontières. Le problème ainsi posé est un problème inverse soluble par différentes méthodes directes ou indirectes (ici un gradient conjugué). On obtient alors le champ de contraintes et de déformations (à différents instants si le problème est dépendant du temps) dans chaque élément constituant cet objet.

2.2 Les limites de l'existant

Les modèles numériques et les modèles analogiques ont chacun leurs avantages et apportent tous leur contribution à la modélisation du phénomène.

2.2.1 Cas particulier des modèles analogiques

Dans une simulation analogique on utilise des matériaux qui ont a priori un comportement proche de celui de la lithosphère, et on leur applique

4. La rhéologie est l'étude du comportement des matériaux soumis à une contrainte.
5. Le mailleur MODULEF de l'INRIA est d'ailleurs mentionné dans la bibliographie.

des conditions aux limites variées. Comme le constate [Jolivet, 1995], le comportement rhéologique des matériaux utilisés ne peut être contrôlé totalement et une part importante de "hasard" entre en jeu. En particulier, des instabilités locales vont souvent être amplifiées et localiser la déformation durant toute l'expérience.

En fait, un modèle analogique peut être envisagé sous deux approches différentes :

- reproduire les déformations naturelles en se focalisant sur des exemples régionaux. Il s'agit alors de comparer les résultats expérimentaux à l'image réelle en tenant compte de l'échelle ;
- étudier un phénomène géotectonique particulier (le plissement, l'érosion, la fracturation...) en faisant varier de façon systématique (comme dans tout travail expérimental) les facteurs régissant la déformation des matériaux de laboratoire utilisés.

Dans les faits, on conjugue souvent ces deux approches et la similitude du modèle et de la réalité suffisent alors à valider le choix des paramètres physiques utilisés sans pour autant en comprendre la raison, même si les auteurs ne le reconnaissent pas toujours [Jolivet, 1995]. On recense deux principales limites à ces modèles expérimentaux : les transferts d'échelles d'une part et les aspects thermiques d'autre part. Ainsi, il est en général impossible de mettre à l'échelle simultanément le temps, les dimensions horizontales et verticales, la rhéologie...et il est donc difficile de se faire une idée précise des rapports entre les structures créées au cours de l'expérience et le problème géotectonique posé. Pour ce qui est des aspects thermiques, il est très difficile de réaliser des modèles expérimentaux incluant un gradient de température au sein de chacune des couches considérées de la lithosphère. L'ensemble du modèle est donc souvent à la même température ce qui constitue une réelle différence d'avec la réalité. Enfin, [Malavieille et al., 1993] note que l'appareillage expérimental employé ne permet pas de prendre en compte les phénomènes de subsidence associés à la compensation isostatique. Cet article répertorie aussi un certain nombre de simplifications de l'approche expérimentale en comparaison des exemples naturels, et remarque notamment que les paramètres géodynamiques tels que l'angle de convergence, la vitesse de convergence et le pendage du plan de subduction restent constants durant toute l'expérience.

2.2.2 Cas particulier des modèles numériques continus

Dans chaque simulation numérique, on suppose connues les propriétés fondamentales de la lithosphère (sa viscosité, ses constantes thermiques...) et on les introduit sous forme de paramètres dans le calcul. En faisant varier les conditions aux limites, on peut tester différentes hypothèses sur la cinématique et les propriétés rhéologiques de la lithosphère.

Les modèles numériques supposent souvent que la lithosphère à un comportement fluide[6]. Comme le remarque [JOLIVET, 1995], dans de tels modèles on observe pourtant des "différences majeures" entre la réalité et le résultat après calcul. Ces écarts sont, d'après lui, imputables au fait que, dans la réalité, les conditions aux limites sont très asymétriques. Par ailleurs, il s'impose que dans le cas numérique, le rôle joué par des structures bien définies ne peut absolument pas être pris en compte. Par contre, ils peuvent donner des indications à très petites échelles (c'est-à-dire vus de très loin) sur la taille des domaines affectés par l'épaississement ou l'extension et les quantités d'épaississement ou d'amincissement fini que l'on peut attendre sur des périodes de temps géologiques sans rencontrer le problème de la mise à l'échelle des modèles analogiques.

Enfin, pour conclure, [JOLIVET, 1995] explique que "si le géologue veut tester le rôle des structures qu'il voit sur le terrain telles que les grands décrochements, seuls les modèles analogiques sont pour l'instant adéquats".

2.3 De l'idée d'une modélisation discrète

Bien que les modélisations décrites ci-dessus soient toujours d'actualité et perpétuellement améliorées, nous avons choisi d'adopter une attitude tangente et de profiter ainsi des acquis de l'analogique tout en restant dans un contexte numérique[7]. Plus simplement, partant de la notion de modélisation analogique en "boite à sable" et l'associant avec les travaux déjà menés dans le domaine de la physique de l'état granulaire [BAK et al., 1987, TANG and BAK, 1988, JAEGER and NAGEL, 1992, GRUMBACHER et al., 1993, DURAN, 1995, MÜLLER, 1996, DURAN, 1997], nous avons décidé d'orienter nos travaux sur une modélisation informatique par système dynamique discret du processus géotectonique.

En effet, même s'il peut sembler relativement "osé" d'associer la rhéologie d'une plaque lithosphérique à de la physique de l'état granulaire, nous justifions notre choix de modélisation de la façon suivante :

- les automates cellulaires (par le biais des méthodes dites de "gaz sur réseau" qui sont une version "microscopique" - mouvement de particules - de l'équation "macroscopique" de Navier-Stokes - écoulement d'un fluide) ont déjà permis de modéliser la convection [LAFAURIE, 1995],

6. On considère souvent dans ce cas que la lithosphère est un milieu continu où la déformation ne se localise pas, comme un fluide à très forte viscosité En fait on estime que les zones déformées sont si denses que l'ensemble se comporte comme un fluide quand il est vu à très petite échelle, c'est-à-dire, de très loin.

7. Pour toutes les raisons énumérées dans le chapitre d'introduction.

- les automates cellulaires ont déjà été utilisés dans des modélisations en sismologie [Narkounskaia and Turcotte, 1992, Huang et al., 1992],
- les automates cellulaires ont déjà été utilisés pour modéliser les mécanismes d'érosion de terrain [Smith, 1991],
- les automates cellulaires (par le biais du Sand Pile Model, qui est un modèle simple d'écoulement de grains dans un espace modélisé par un graphe) sont utilisés pour modéliser les phénomènes tels que les avalanches [Durand-Lose, 1996][8],
- dans l'univers "réel", les interactions sont locales (écoulements, avalanches, propagation...), limitées par la vitesse de la lumière et uniformes dans l'espace,
- au cours d'un processus géotectonique comme la subduction, plusieurs phénomènes physiques et chimiques concurrents se produisent sur des échelles de temps bien distinctes et portent sur des couches de terrain très hétérogènes qui ne peuvent absolument pas être considérées comme un tout. Adopter une approche où chaque portion de l'espace étudié est considérée dans sa singularité pour une échelle de temps donnée[9] nous semble donc bien être un bon élément de réponse.

Or, comme nous pourrons le constater en recensant les phénomènes à "reproduire" sur la portion "d'espace" que nous devons modéliser (portion limitée aux pourtours de la fosse océanique), il faut essentiellement simuler des effets d'affaissements de terrain, d'avalanches superficielles, d'érosion sous-crustale...Et tous ces phénomènes ont déjà été plus ou moins abordés, individuellement, par une technique de modélisation discrète. Notre approche consiste donc en une tentative d'intégration de ces différents modèles en un seul et unique, afin de simuler un ensemble varié de phénomènes.

Enfin, nous avons constaté que les modèles numériques continus ont été longuement et soigneusement développés et améliorés sans pour autant répondre aux attentes de l'ensemble de la communauté des géologues et géophysiciens. C'est aussi pour cette dernière raison que nous avons choisi de nous démarquer de la tendance générale.

8. Ces phénomènes sont appelés "1/f phenomenon" car ils vérifient statistiquement un rapport inverse entre les intensités des perturbations et leurs probabilités. Ils ont aussi une propriété de criticalité auto-organisée, ce qui signifie qu'ils ont la propriété de se retrouver immédiatement après un bouleversement dans un état pseudo-stable susceptible de se re-bouleverser instantanément.

9. La notion d'échelle de temps dans le sens où nous l'employons sera approfondie dans la suite de ce mémoire.

Chapitre 3

Ordres de grandeur et restrictions de modélisation

Au cours de ce bref chapitre, nous allons préciser la problématique de notre modélisation ; c'est-à-dire que nous exposerons nos hypothèses de travail relativement restrictives et leurs limites. Commençons d'abord par présenter la géométrie d'ensemble de notre "plan de coupe de modélisation" et sa dynamique, énumérons ensuite les phénomènes à reproduire avant d'aborder la notion des échelles de temps.

3.1 Géométrie d'ensemble

Comme nous l'avons vu au cours du chapitre de description de la tectonique des plaques, au fur et à mesure de la dérive de part et d'autre d'une dorsale, la lithosphère nouvellement créée se refroidit peu à peu. Son niveau d'équilibre isostatique, c'est-à-dire son niveau de flottaison sur l'asthénosphère, s'enfonce alors corrélativement [Jolivet, 1995]. La surface de la plaque océanique qui s'écarte donc, subside et la profondeur du fond augmente.

3.1.1 Les "grands ensembles"

En ce qui concerne notre propre modèle, nous avons restreint l'étude de la subduction aux pourtours de la fosse océanique. En fait, nous limitons la portée de notre modèle à une dizaine de kilomètres en amont et en aval de la fosse. Ainsi, si nous considérons une zone de subduction type [1], où la fosse océanique est située à une profondeur d'environ 6 km, la lithosphère océanique plonge sous la lithosphère continentale avec un pendage superficiel de

1. Comme celle qui se trouve au large du Pérou par exemple. Le lecteur pourra se reporter avec profit au récapitulatif concernant la géométrie des marges fait dans [Schnürle, 1994].

plus de 7°. Ce pendage augmente au delà d'une quinzaine de kilomètres, mais ceci sort déjà du cadre que nous nous sommes fixé. Le premier grand ensemble à prendre en compte correspond donc à la lithosphère en subduction qui passe d'un niveau quasi sub-horizontal à l'avant de la fosse, à un pendage quasi constant sous l'arc volcanique. Dans la pratique, et pour adapter le modèle à la réalité, nous avons fixé la valeur des paramètres de pendage de la plaque inférieure à : 2,25° en amont de la fosse (il s'agit pour le modèle d'une zone longue de moins de 10 km, appelée mur externe*), 3,6° sous la fosse et 8,1° sous la plaque chevauchante.

Le deuxième grand ensemble à représenter correspond à la fosse océanique. Nous avons arbitrairement choisi de considérer une fosse type de 10 km de large et 800 m dans sa plus grande profondeur (c'est-à-dire au niveau du "chenal de subduction").

Le troisième grand ensemble à représenter correspond à la plaque chevauchante (encore appelée marge continentale). Surplombant la fosse océanique et chevauchant la plaque inférieure en subduction, la base de cette plaque permet de délimiter le chenal de subduction. Ce chenal (qui a une réelle existence physique et qui est le lieu de hautes pressions [Bourgois, 1993]) est considéré, dans notre modèle, comme un parallélépipède dont la base est délimitée par la plaque inférieure. Il est bordé par la fosse océanique sur l'un des cotés et est appelé à se déformer au cours de la simulation. Le pendage de la base de la plaque supérieure est donc, pour ce qui est des conditions initiales du moins, égal au pendage du chenal de subduction, c'est-à-dire égal au pendage de la plaque en subduction (8,1°). Alors que le pendage de la partie supérieure de la plaque chevauchante, encore appelée mur interne*, est proche de 6°(valeur moyenne).

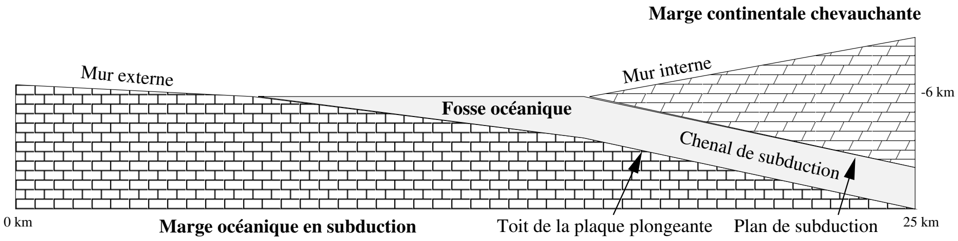

Fig. 3.1 – *Plan de coupe simplifié*

Pour terminer cette description des grands ensembles, il ne reste plus qu'à mentionner l'océan qui borde la partie superficielle supérieure des deux lithosphères et de la fosse.

3.1.2 Dynamique et moteurs

De notre point de vue, la cinématique de ces marges est assez simplifiée par le fait que nous ne prenons en compte que la progression oblique descendante, à vitesse constante, de la plaque en subduction. La lithosphère chevauchante, parce qu'elle constitue le référentiel, reste immobile. De plus, pour reprendre une expression de [Schnürle, 1994], nous retiendrons l'idée d'un modèle isostatiquement compensé où les contraintes "extérieures" imposées aux bords peuvent être surimposées. Pour nous, l'accrétion ou l'érosion de matériel tectonique à la marge continentale, peut être assimilées, en première approximation, à une augmentation ou une diminution de la charge pondérale que supporte la marge en subduction.

Partant du principe que le plongeon de la plaque en subduction est "immuable" et qu'il se produit à vitesse constante, nous avons cherché un mécanisme pour représenter l'érosion profonde qui soit indépendant des fluctuations de la cinématique même de la subduction. Nous avons écarté l'effet érosif de la subduction d'aspérités de la croûte océanique, car, comme le note [Lallemand and Malavieille, 1992], les reliefs sous-marins, s'ils peuvent agir comme des accélérateurs temporaires de l'érosion, ne peuvent en aucun cas en être le moteur principal. Ils ne suffisent en effet en aucune façon à rendre compte de la totalité du volume entraîné par subduction-érosion.

En fait, pour modéliser l'érosion, nous avons repris l'idée d'une forte influence des fluides sous pression contenus dans les sédiments qui pénètrent dans le chenal et provoquent une fracturation hydraulique de la base de la plaque chevauchante sus-jacente et donc son érosion. Plus pratiquement, cette hydrofracturation est représentée dans notre modèle, par le franchissement d'une valeur seuil pour un attribut dit "de vieillissement". Cet attribut, propre à chacune des cellules est susceptible d'être augmenté ou non pseudo aléatoirement à chaque pas de temps en fonction de la nature de son voisinage et des efforts qui lui sont appliqués.

Pour en terminer avec les moteurs de notre modèle, il nous reste encore à aborder l'ensemble des effondrements gravitaires. Ils peuvent être de plusieurs types : décrochement sous-crustal de portions de terrain de la base de la plaque chevauchante (suite à un concours de conditions telles que le franchissement de la valeur seuil précitée pour le coefficient de vieillissement de la cellule) en vue d'un charriage en profondeur des éboulis, affaissement de terrain du fait de la formation de cavités trop importantes à la base de la plaque supérieure (sous l'effet d'une surcharge pondérale), avalanches superficielles au sommet de la plaque supérieure...

3.2 Phénomènes à reproduire

Au large du Pérou, la fosse océanique mesure une dizaine de kilomètres de large. La pente supérieure de la marge continentale varie entre 12% (5,4°) et 28% (12,6°). Le pendage entre l'horizontale et le plan de subduction est de 17% (7,65°). La plaque océanique plonge à la vitesse de 5 cm/an, tandis que la plaque continentale se rapproche du centre de la fosse océanique (c'est-à-dire encore qu'elle converge elle même vers la plaque océanique) à la vitesse de 3 cm/an. De façon générale, un repère fixe placé sur la plaque océanique "donne l'impression" de plonger à une vitesse de 80 km/Ma. Le point de contact situé au front de la marge chevauchante recule de 20 km/Ma, il garde cependant une "altitude" constante.

La superposition de l'ensemble de nos "moteurs" doit entraîner les effets suivants : une rupture de pente de la plaque plongeante en proportion de l'effort pondéral exercé par la plaque chevauchante, un crénelage approximatif de la plaque plongeante (résultant des brisures de pente successives), un sous-charriage des matériaux érodés, une "impression" d'hydro-fracturation. Il faut bien sûr aussi respecter les contraintes suivantes : rendu visuel avec effets de "brisure" ou ruptures de terrain dans la géométrie de la plaque chevauchante et recul simultané du front de la marge chevauchante et de l'axe de la fosse océanique à altitude constante.

3.3 Les échelles de temps

Pour mettre en place l'ensemble des "moteurs" que nous avons déjà mentionné, nous avons dû faire appel à la notion d'échelles de temps. Ces échelles de temps sur lesquelles nous reviendrons dans les chapitres suivants correspondent en fait à l'introduction de contextes bien distincts. Ainsi, à titre d'exemple, nous avons estimé que nous ne pouvons pas décemment mettre un phénomène aussi "instantané" qu'un éboulement de terrain à égalité avec un phénomène plus difficilement perceptible le temps d'une vie humaine comme l'est la subduction d'une plaque. Nous avons par conséquent cherché à regrouper par classe les "moteurs" opérant au niveau des mêmes échelles de temps.

Deuxième partie

Les modèles dynamiques discrets

Chapitre 4

Le formalisme "automates cellulaires"

Comme [Ferber, 1995] le constate :

> *"notre intuition commune appréhende l'action comme une modification locale : mes actes ne transforment que les entités qui se trouvent proches de moi. Si je fais tomber un verre par terre, je ne suppose pas que le cours des planètes s'en trouvera modifié. Toute action ne produit qu'une perturbation locale, une altération qui se trouve en contact ou de toute manière à une distance finie de la cause de cette action."*

Partant de ce constat, nous avons choisi de modéliser le phénomène de subuction-érosion en tectonique des plaques par une approche locale. Ainsi, nous considérons que le milieu d'étude (une partie du globe terrestre) est composé d'une quantité dénombrable d'éléments (des portions d'espace) qui interagissent à un niveau local lorsqu'ils sont stimulés.

Toute la complexité de cette modélisation réside dans la représentation des différentes échelles des grandeurs concernées et dans l'établissement des lois d'évolution.

4.1 Introduction aux réseaux d'automates cellulaires

En première approche, un réseau d'automates cellulaires peut-être défini comme un ensemble de points, appelés cellules, possédant chacun une valeur (un état) d'un ensemble fini d'états. A chaque itération, tous les états sont remis à jour simultanément grâce à une unique fonction de transition dépendant d'un même voisinage relativement à la cellule concerné. La dynamique est purement locale, uniforme, sans interactions à longue distance.

Du point de vue de la terminologie, la littérature parle de *réseau d'automates cellulaires* ou éventuellement d'*automates cellulaires* (au pluriel) pour désigner l'ensemble des cellules. L'appellation *automate cellulaire* (au singulier) désigne une cellule du réseau d'automates cellulaires.

4.1.1 Le système discret

[Vichniac, 1984, Margolus, 1984, Wolfram, 1986, Wolfram, 1994], à propos de la topologie des réseaux d'automates cellulaires utilisés, parlent de grilles régulières ou tableaux de cellules. Pour [Durand, 1994], on peut trouver des définitions plus générales des automates cellulaires en les faisant agir, non dans $\mathbb{Z}^n$, mais sur un graphe de Cayley. Ainsi, d'après [Róka, 1994], un réseau d'automates cellulaires est formé de la juxtaposition d'un même automate fini placé sur les sommets de graphes plus complexes qu'une grille ou $\mathbb{Z}^n$.

Pour nos deux modèles qui seront présentés au cours des chapitres suivants, nous avons décidé de discrétiser la coupe verticale correspondant à la zone de subduction (voir fig. 3.1), par des grilles régulières de deux façons distinctes. La modélisation se fait donc dans un plan vertical. La première différence entre ces deux modèles réside donc dans la définition physique, même, des éléments de base. Nous appellerons dorénavant *cellules* ces entités élémentaires indivisibles correspondant aux sommets des graphes réguliers que nous avons choisis pour "mailler" la coupe.

Le plan de coupe ayant été représenté symboliquement par une grille régulière, il nous faut aussi lui associer un état initial. Cet état de départ que l'on définit arbitrairement, correspond à la représentation symbolique d'un état physique du plan de coupe de la subduction à une date donnée.

4.1.2 Le mode opératoire ou la dynamique du système

Une fois la géométrie et l'état initial du réseau définis, il faut ensuite faire évoluer globalement notre système au cours du temps. Pour ce faire, nous commençons par discrétiser la grandeur "temps". Chaque pas de temps est ainsi marqué par un pas d'horloge[1]. Puis, comme l'évolution globale ne dépend que des interactions locales, il faut préciser les règles d'influences mutuelles entre les cellules (nous parlerons par la suite de *fonctions de transitions locales*).

Ces principes étant posés, il ne reste plus qu'à laisser le système évoluer de lui-même, par réévaluation de l'état de chacune des cellules en fonction de ceux de ses voisines[2], à chaque pas d'horloge [Heudin, 1994, Hillis, 1988]. Cette réévaluation des états de chacune des cellules de la subdivision

1. Il importera plus loin de définir une échelle de temps prenant en compte la diversité des phénomènes étudiés.

2. Au sens où elles influencent la cellule courante...

(du maillage), peut se faire selon un mode synchrone (toutes les cellules changent d'état en parallèle) ou non.

Nous venons ainsi de définir une *dynamique*[3] (un univers de comportement) à temps discret pour notre système discret. Nous nous retrouvons donc exactement dans un cas d'application des *réseaux d'automates* ([ROBERT, 1995]).

4.2 Le formalisme utilisé

4.2.1 Les réseaux d'automates

Le système discret comme ensemble de cellules Soit S un ensemble dénombrable, non nécessairement fini, de cellules (entités élémentaires introduites en 4.1.1). Toutes les cellules de S peuvent prendre un état appartenant à un même ensemble fini E.

Une configuration* de l'ensemble S des cellules, est une application c de $S \mapsto E$, qui associe à chaque cellule de l'ensemble S son état. L'ensemble de toutes les configurations possibles de S est E^S.

Une dynamique ou fonction de transition globale* sur S est une application F de $E^S \mapsto E^S$ exprimant pour l'ensemble des cellules de S, le passage d'une configuration à une autre configuration. Appelons p_s la projection selon la composante $s \in S$, de E^S sur E ; $p_s \circ F$ est une application de $E^S \mapsto E$ appelée fonction de transition locale* associée à la fonction F. Au sein de l'ensemble des fonctions de transition globales, on distingue le cas particulier des *fonctions de transition globales série-parallèles*:

Fonctions de transition globales série-parallèles F est une fonction de transition globale série-parallèle sur S, si et seulement s'il existe une partition $(P_1, \ldots, P_m)$ finie d'ordre m ($m \in \mathbb{N}$) de l'ensemble S, et une suite d'applications $(h_i)_{i \in 1,\ldots,m}$ de $E^S \mapsto E^S$ vérifiant, pour tout i de $1, \ldots, m$:

$$\forall s \in S, \;\; p_s \circ h_i = \begin{cases} p_s \circ F & si \;\; s \in P_i \\ p_s & si \;\; s \in S \setminus P_i \end{cases}$$

telles que $F = h_m \circ \ldots \circ h_1$. Comme cas particuliers de fonctions de transition globales série-parallèles, nous pouvons définir les *fonctions de transition globales purement parallèles* (cas où $m = 1$) et les *fonctions de transition*

3. D'après [BOURRET et al., 1991], un *système dynamique autonome*, homogène, c'est la donnée d'un ensemble des temps $\mathcal{T}$ (discret ou continu), d'un espace d'états E (qui peut-être discret ou un domaine géométrique régulier d'un espace de dimension finie) et d'une fonction d'évolution du système $F : \mathcal{T} * E \mapsto E$ telle que l'état du système à la date $t \in \mathcal{T}$ puisse être calculé en fonction de l'état du système à une date antérieure $t_0 \in \mathcal{T}$ et de l'intervalle de temps $t - t_0$. Une telle définition implique que $\forall t, t' \in \mathcal{T}, \forall c \in E$, $(t' \geq t) \Rightarrow F(t', c) = F(t' - t, F(t, c))$.

globales purement séries (dans ce dernier cas, il faut que S soit fini et que $m = |S|$, où $|S|$ désigne le cardinal de l'ensemble S).

Notion d'influence relative à une fonction de transition globale donnée Soient q et r deux cellules de S. On dit que la cellule q influence* la cellule r *relativement à la fonction de transition globale* F si et seulement s'il existe $c_1, c_2 \in E^S$ deux configurations de l'ensemble des cellules S telles que :

$$\forall s \in S \setminus q, \quad c_1(s) = c_2(s)$$

pour lesquelles : $p_r \circ F(c_1) \neq p_r \circ F(c_2)$. Note : nous avons donc alors nécessairement $c_1(q) \neq c_2(q)$.

Construction du graphe d'influence de F Cette notion d'influence[4] relative à une fonction de transition globale F permet d'introduire une relation* sur S, c'est-à-dire encore un *graphe simple* orienté $G_F = (S, A_F)$. Le couple $(q, r) \in S * S$ constitue un arc du graphe G_F (c'est-à-dire un élément de l'ensemble A_F), si et seulement si la cellule q influence la cellule r relativement à la fonction de transition globale F. L'ensemble des cellules *adjacentes*[5] (au sens des graphes) à la cellule $s \in S$, est l'ensemble $A_F(s) \cup A_F^{-1}(s)$ des cellules ayant une influence sur s ou influencées par s, relativement à F.

Cellule active relativement à une fonction de transition globale donnée Une cellule $s \in S$ est inactive* pour la fonction de transition globale F si et seulement si $p_s \circ F = p_s$. Une cellule qui n'est pas inactive pour F est appelée cellule active* relativement à la fonction de transition globale F.

Temps discret - principe des itérations discrètes - modes opératoires Il s'agit à présent de définir un "espace des temps" $\mathcal{T}$ discret, muni d'une loi de composition interne (additive), d'une relation d'ordre total et d'un plus petit élément (l'origine des temps, élément neutre pour l'addition). Pour des raisons pratiques, nous le choisissons égal à $p\,\mathbb{N}$ (avec $p \in \mathbb{N} \setminus \{0\}$, le *pas de temps*) et plaçons l'origine des temps en 0.

On dit d'une suite de configurations de l'ensemble des cellules de S, qu'elle suit le principe des itérations discrètes sur $\mathcal{T}$, si et seulement si cette suite est indexée par $\mathcal{T}$, c'est-à-dire encore si et seulement si c'est une application de $\mathcal{T} \mapsto E^S$.

4. Nous pourrions préciser "d'influence immédiate" étant donné que nous allons introduire dans la suite de ce document la notion d'échelle de temps.

5. C'est-à-dire, l'ensemble des cellules distinctes de s pour lesquelles il existe un arc les reliant à s, indépendamment de l'orientation.

Un mode opératoire associé à une suite de configurations $(c_t)_{t\in\mathcal{T}}$ de l'ensemble des cellules de S, est une suite de fonctions de transition globales $(F_t)_{t\in\mathcal{T}}$ exprimant le passage du système à la configuration $c_{succ(t)}$, pour toute date $t \in \mathcal{T}$. On distingue trois familles particulières de modes opératoires :

- les *modes opératoires série-parallèles* (homogènes)

  Pour ces modes opératoires, la suite des fonctions de transition globales est constante, égale à une fonction de transition globale série-parallèle F.

- les *modes opératoires asynchrones*

  Pour ces modes opératoires, il faut définir une suite $(S_t)_{t\in\mathcal{T}}$ d'éléments de $\mathcal{P}(S)$ (des sous-ensembles de S) regroupant les cellules actives au cours du temps et une suite de temps de propagation $(\Theta_t)_{t\in\mathcal{T}}$ (des éléments de $\mathcal{T}^{S*S}$) des influences sur le graphe[6]. Alors, à chaque date $t \in \mathcal{T}$, le calcul de la configuration $c_{succ(t)}$ se fait de la manière suivante :

  - si $s \in S \setminus S_t$, alors $c_{succ(t)}(s) = c_t(s)$,
  - si $s \in S_t$, alors $c_{succ(t)}(s) = p_s \circ F_t(c'_{t,s})$, avec

  $$c'_{t,s} : \begin{array}{l} S \mapsto E \\ s' \to c_{t-\Theta_t(s',s)}(s') \end{array}$$

  Pour ces modes opératoires, le calcul des configurations ne porte que sur un sous-ensemble de cellules dites "actives" (dans le sens où elles sont seules à être réévaluées à ce pas de temps donné), et l'on introduit par ailleurs une notion de temps de retard qui correspond au temps de propagation des influences sur les arcs du graphe.

- les *modes opératoires chaotiques*

  Ce sont des cas particuliers de modes opératoires asynchrones dépourvus de la notion de temps de retard pris à la propagation des influences sur les arcs du graphe. C'est-à-dire encore, que la suite de temps de propagation $(\Theta_t)_{t\in\mathcal{T}}$ (des éléments de $\mathcal{T}^{S*S}$) des influences sur le graphe est constante égale à 0_{S*S}.

Les réseaux d'automates On définit un réseau d'automates* par la donnée d'un quintuplet $(G, E^S, (F_t)_{t\in\mathcal{T}}, (S_t)_{t\in\mathcal{T}}, (\Theta_t)_{t\in\mathcal{T}})$ où S désigne un ensemble dénombrable de cellules, E désigne un même ensemble fini d'états que peuvent prendre chaque cellule de S, $(F_t)_{t\in\mathcal{T}}$ désigne un mode opératoire

6. Par convention : si $(q,r) \notin A_{F_t}$, alors $\Theta_t(q,r) = 0$. Il n'y a pas de retard de pris à la propagation d'une "non-influence".

de S sur $\mathcal{T}$, $(S_t)_{t\in\mathcal{T}}$ désigne la suite des cellules actives au cours du temps et $(\Theta_t)_{t\in\mathcal{T}}$ la suite des temps de propagation des influences sur le graphe G. G est le graphe simple orienté défini par : $G = (S, \bigcup_{t\in\mathcal{T}} A_{F_t})$.

4.2.2 Les réseaux d'automates cellulaires

Dans le modèle réseau d'automates, il est possible d'avoir des règles de transition et des types de voisinages différents d'une cellule à l'autre et différents au cours du temps. Nous allons, à présent, restreindre cette définition au cas des *réseaux d'automates cellulaires* pour lesquels il s'agit de juxtaposer une même règle de transition selon un mode opératoire synchrone homogène en des cellules qui ont toutes le même type de voisinage (indépendamment du temps).

Avertissement Dans [Róka, 1994], on peut lire :

> La première notion apparue pour simuler de tels réseaux est celle de graphe d'automates présentée par Rosenstiehl et al [...] On part alors d'un graphe de degré fixé ; sur chaque sommet, on place un automate fini qui communique avec ses voisins par les arêtes du graphe. Mais cette notion n'est pas encore satisfaisante : une longue série de travaux montre qu'il est nécessaire d'introduire une notion de "vecteur de repérage" pour faire fonctionner le réseau : si chaque machine connaît ses voisines suivant un certain ordre local figurant sur les arêtes du graphe, elle doit aussi connaître son propre numéro dans les listes de chacune de ses voisines. Comme on veut modéliser des réseaux construits uniformément, il est gênant de passer par un tel vecteur de repérage : on devrait déduire "naturellement" l'information qu'il contient ; on va donc se restreindre à une classe de graphes plus réguliers.

Soit $(G, E^S, (F_t)_{t\in\mathcal{T}}, (S_t)_{t\in\mathcal{T}}, (\Theta_t)_{t\in\mathcal{T}})$ un réseau d'automates. Comme ce qui caractérise un réseau d'automates cellulaires est l'existence d'un réseau régulier et uniforme de machines finies s'influençant mutuellement, il faut commencer par imposer aux graphes $G_t = (S, A_{F_t})$ d'être connexes[7] et d-réguliers[8] avec $d \geq 1$, pour $t \in \mathcal{T}$ (si l'on veut s'attacher à définir les réseaux d'automates cellulaires du moins). Ensuite, puisque le mode opératoire du réseau doit-être homogène, tous les termes de la suite de fonctions $(F_t)_{t\in\mathcal{T}}$ doivent-être identiques et égaux à une même fonction de transition globale F indépendante du temps t. C'est aussi pour cette raison que nous considérons

7. Ce qui signifie qu'entre deux cellules quelconques, il existe toujours une chaîne.

8. Un graphe (S,A) est d-régulier si $\forall s \in S, |A(s)| = d$. Ici, ceci signifie que toute cellule influence un même nombre de cellules.

désormais qu'il n'y a pas de notion de temps de retard pris par la propagation des influences sur les arcs du graphe. La suite $(\Theta_t)_{t\in\mathcal{T}}$ est donc constante égale à 0_{S*S}.

Enfin, pour toutes les cellules $s \in S$, il faudrait pouvoir écrire $p_s \circ F$ sous la forme d'une seule et même fonction de transition locale $f : E^S \mapsto E$ indépendamment de la cellule considérée. Or, en fait, seuls les états des cellules de l'ensemble $A^{-1}(s)$ importent réellement à l'évaluation de $p_s \circ F(c)$ (pour $c \in E^S$).

Ce problème étant loin d'être trivial dans le cas général où l'on ne possède pas plus d'informations sur la fonction de transition globale F, l'influence A_F et l'ensemble des cellules S, nous allons nous "restreindre à une classe de graphes plus réguliers".

Définition Soit S un ensemble infini dénombrable de cellules placées chacunes en un point de $\mathbb{Z}^n$. Soit $(G, E^S, (F_t)_{t\in\mathcal{T}}, (S_t)_{t\in\mathcal{T}}, (\Theta_t)_{t\in\mathcal{T}})$ un réseau d'automates à mode opératoire purement parallèle, synchrone, homogène sur cet ensemble S de cellules, où, pour tout $t \in \mathcal{T}$, $G_t = (S, A_{F_t})$ est un même graphe simple orienté, connexe et d-régulier (noté $\mathcal{G}$). Appelons F l'unique fonction de transition globale, indépendante du temps. On a donc $\forall t \in \mathcal{T},\ \ A_F = A_{F_t}$ et, par ailleurs, $G = (S, A^\infty) = (S, \bigcup_{t\in\mathcal{T}} A^t)$.

Soit σ la cellule origine des axes, posons $V = A_F^{-1}(\sigma)$ l'ensemble des cellules de S ayant une influence sur σ. Le graphe vérifie alors la propriété suivante d'invariance par translation : $\forall s \in S,\ \ A_F^{-1}(s) = s + V$. V est appelé l'adjacence locale relative.

Ainsi, pour toute cellule $s \in S$, $p_s \circ F$ est évaluée à partir de la donnée de V, et peut donc être écrite sous la forme d'une application $f : E^d \mapsto E$.

Cas particulier où S est fini On complète l'ensemble S par une quantité infinie dénombrable de cellules inactives et à état neutre[9] pour la fonction de transition locale f.

Définition On appelle réseau d'automates cellulaires* le quadruplet $(\mathcal{G}, E^S, V, f)$ où $\mathcal{G} = (S, A)$ est un graphe simple orienté, connexe et d-régulier, E est l'ensemble fini des états que peuvent prendre les différentes cellules, V l'adjacence locale relative et f la fonction de transition locale*. Tous ces éléments ayant été définis précédemment.

9. Dans la pratique, un tel état peut-être obtenu par ajout d'un attribut spécifique supplémentaire à chacune des cellules.

Chapitre 5

Le modèle uni-dimensionnel

Pour établir un modèle par réseau d'automates cellulaires uni-dimensionnel du phénomène géotectonique, nous nous sommes fortement inspirés des principes du Sand Pile Model* de [Bak et al., 1987]. Au cours de ce chapitre, après avoir présenté ce modèle, nous exposerons donc les principes qui nous ont amené à le généraliser puis nous aborderons la description de notre propre réseau d'automates cellulaires en commençant par décrire le "pavage du plan", l'ensemble des états d'une cellule, son voisinage, la fonction de transition locale et la configuration initiale. Nous présenterons aussi l'algorithme simplifié correspondant.

5.1 Le tas de sable

A la fin des années 1980, P. Bak, C. Tang et K. Wiesenfeld du Brookhaven National Laboratory (États-Unis) ont développé un modèle cherchant à reproduire un mécanisme d'avalanche dans le cadre de la physique de l'état granulaire.

La dynamique des tas de sable*, ou Sand Pile Model* (SPM), est basée sur l'évaluation de la différence de hauteur de grains entre deux sites (ou piles) voisins. Le système correspond à une ligne infinie de piles de sable représentant une coupe verticale du tas de sable. Chaque pile est représentée par un entier naturel correspondant au nombre de grains qui la composent. Le tas de sable est donc modélisé dans son ensemble par une suite infinie d'entiers naturels. A chaque instant et simultanément avec l'ensemble des piles du tas, chaque pile évalue sa propre hauteur de grains relativement à ses deux voisines immédiates. Si l'on adopte la loi de transition décrite dans [Durand-Lose, 1996], à chaque pas de temps et concurremment, une pile perd un grain au profit de chacune des colonnes voisines qui a au moins deux grains de moins qu'elle et prend un grain à chacune de ses voisines qui a au moins deux grains de plus qu'elle.

Cette règle de transition est déterministe, mais ce n'est pas toujours le cas ([MÜLLER, 1996] donne un exemple de loi de transition probabiliste). La dynamique de ce réseau d'automates cellulaires est à mode opératoire purement parallèle, la fonction de transition locale f s'écrit simplement. Ainsi, si $C^t = (C^t_j)_{j\in\mathbb{Z}}$ désigne la suite infinie d'entiers naturels représentant le tas de sable à une date t donnée ($t \in \mathbb{N}$). Alors, on a :

$$\begin{array}{lll} f: & \mathbb{N} * \mathbb{N} * \mathbb{N} & \mapsto \mathbb{N} \\ & (C^t_{j-1}, C^t_j, C^t_{j+1}) & \to C^{t+1}_j \end{array}$$

avec : $C^{t+1}_j = C^t_j - \mathbb{I}(C^t_j - C^t_{j-1}) - \mathbb{I}(C^t_j - C^t_{j+1}) + \mathbb{I}(C^t_{j-1} - C^t_j) + \mathbb{I}(C^t_{j+1} - C^t_j)$

$$\text{et} : \mathbb{I}(n) = \begin{cases} 0 \text{ si } n < 2 \\ 1 \text{ sinon} \end{cases}$$

Chaque pile peut accueillir une quantité, non bornée a priori, de grains, mais il doit y avoir un nombre fini - et constant au cours du temps - de grains dans le système.

Dans ce modèle, l'espace est un quadrillage à mailles carrées que l'on implémente sous la forme d'un tableau uni-dimensionnel de longueur infinie. La représentation graphique associée est une image à deux dimensions, mais le réseau d'automates cellulaires considéré est à une dimension car le tas de sable est assimilé à une suite de hauteurs de piles de sable.

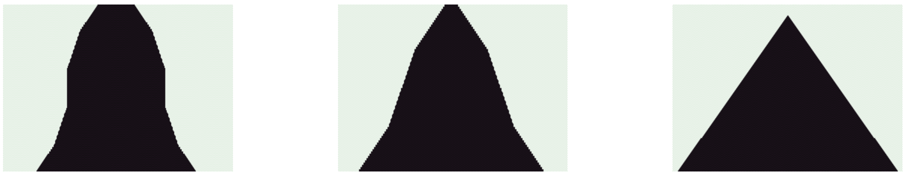

FIG. 5.1 – *Copies d'écran du Sand Pile Model uni-dimensionnel*

Comme le note [MÜLLER, 1996], la caractéristique fondamentale d'un tel modèle est que les lois de la physique n'interviennent pas en tant que telles. Cependant les règles d'évolution sont choisies de façon à reproduire au mieux les lois naturelles.

Dans la réalité cependant, les tas de sable ne se comportent pas exactement selon le modèle de criticalité auto-organisée observé sur les automates cellulaires du SPM. En effet, comme le rapportent [JAEGER and NAGEL, 1992, MÜLLER, 1996], en 1990, Glen A. Held et ses collègues du centre IBM de Yorktown Heights ont construit un appareil expérimental permettant de verser des grains de sable un par un sur un plateau. Leurs expériences ont montré que si le rayon de la base du tas est inférieur à environ trente diamètres de grains, les prédictions du SPM s'avèrent exactes, au delà, ce n'est plus le cas.

5.2 Le système dynamique discret

Pour élaborer un premier modèle du phénomène de subduction-érosion en tectonique des plaques, nous sommes partis des deux simplifications suivantes :

- le "plan de coupe de modélisation" est bien déterminé et de taille fixe ; il est possible en première (et grossière) approximation de le représenter sous forme d'un empilement d'un nombre fini et déterminé de couches successives ;
- comme pour le tas de sable et le Sand Pile Model où la physique n'intervient pas vraiment en tant que telle, nous cherchons à mettre en place un modèle par réseau d'automates cellulaires uni-dimensionnel où les lois de la géotectonique, de la géochimie et de la géophysique n'interviennent pas en tant que telles.

5.2.1 Pavage de la zone et ensemble des états d'une cellule

Pour des raisons pratiques d'implémentation informatique, nous ne pouvons pas manipuler un ensemble de cellules infini. Par ailleurs, il s'agit de modéliser une portion de coupe de zone de subduction de 25 km de long sur 5 km de haut. Notre réseau d'automates cellulaires, fortement inspiré des automates du Sand Pile Model, est donc une ligne finie d'automates. En ce qui concerne les extrémités [1], nous avons choisi de les singulariser par un état "frontière" invariant.

Dans le Sand Pile Model, une cellule représente une verticale du plan de coupe, c'est-à-dire une portion du plan comportant (ou non) une colonne de sable et son complément à la hauteur totale du plan de coupe, de vide "ambiant". Elle est donc "représentable" par un entier naturel correspondant à la hauteur de l'empilement de grains de sable sur une verticale. Pour le modèle de subduction érosion, le nombre de couches à prendre en compte sur une verticale donnée est a priori variable et moins limité. Dans la pratique, nous avons, arbitrairement, décidé de nous limiter à un modèle en huit couches mais il est tout à fait possible d'augmenter ce nombre en étoffant le modèle. Il s'agit de :

- la couche correspondant à la plaque océanique en subduction,
- la couche représentant le matériel décroché de la marge chevauchante et déposé sur la surface de la plaque océanique en subduction,
- la couche intermédiaire entre les deux plaques lithosphériques représentant simultanément (en très grossière approximation) le matériel de la fosse océanique et le matériel du chenal de subduction,

1. Qui dans notre cas pratique représentent $0,2\%$ de l'ensemble des cellules.

– la couche de matériel continental "fortement hydro-fracturé" placé à la base de la plaque chevauchante et susceptible d'être décroché de la plaque supérieure pour être déposé sur la plaque océanique et charrié vers les profondeurs,

– trois couches différentes au sein de la plaque continentale, de coefficient de "friabilité" croissant au fur et à mesure que l'on s'éloigne du front[2] de la plaque supérieure, et de la surface de la plaque continentale,

– la couche d'eau qui correspond au complément à la hauteur du plan de coupe des sept couches précédentes.

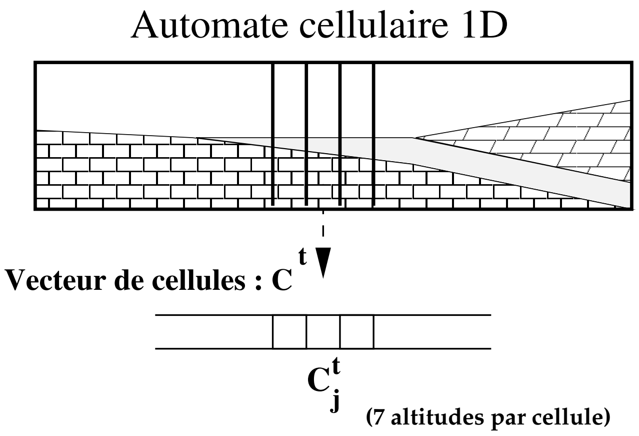

FIG. 5.2 – *Maillage uni-dimensionnel de la zone de subduction*

En plus de ces huit couches (qui seront représentées par sept entiers naturels majorés par la hauteur totale du plan de coupe étudié), l'état d'une cellule est déterminé par la donnée de deux autres coefficients entiers. L'un de ces coefficients est un "drapeau" indiquant ou non la présence d'une marche[3] sur la verticale considérée. Ce coefficient prend ses valeurs dans l'ensemble fini $\{0, \ldots, L_{marche}\}$ où L_{marche} est une constante entière correspondant à la longueur totale[4] d'une brisure du sommet de la plaque océanique. Le second coefficient est appelé coefficient de "vieillissement". Bien qu'il ne corresponde à aucune donnée physique réelle à proprement parler, nous l'avons défini pour permettre d'évaluer le degré de friabilité

2. Le front de la plaque supérieure est le point ou l'ensemble de points de la plaque supérieure, qui jouxtent la fosse océanique.

3. Ces marches, ou brisures, nous permettent d'obtenir un rendu-visuel avec effet de crénelage sur la plaque plongeante, ce qui correspond approximativement aux ruptures de pente successives.

4. Pour des questions de simplification à la mise en œuvre du modèle, nous avons décidé que la longueur de ces "brisures" est une constante.

d'une cellule donnée[5]. C'est la conjonction de ce facteur et de la partie des règles de transition qui le prend en compte qui nous permet de représenter le phénomène réel d'hydrofracturation. Ce facteur est un entier naturel qui prend ses valeurs dans l'ensemble fini $\{0, \ldots, MAX_{vieil}\}$ où MAX_{vieil} est une constante dont la valeur est réglée empiriquement en fonction du rendu visuel général.

L'état d'une cellule du réseau d'automates cellulaires est donc un élément de l'ensemble fini :

$$E = \{0, \ldots, H_{totale}\}^7 \times \{0, \ldots, L_{marche}\} \times \{0, \ldots, MAX_{vieil}\}$$

5.2.2 Fonction de transition locale : nécessité d'un mécanisme d'imbrication

Comme nous l'avons déjà écrit au cours des précédents chapitres, pour modéliser une convergence de marge avec phénomène d'érosion, il faut représenter trois grandes "tendances" distinctes : un plongeon de la plaque océanique, une érosion de la base de la plaque continentale accompagnée des affaissements de terrain qui en résultent sur toute la hauteur de la plaque continentale et des avalanches superficielles au sommet de la plaque continentale. Ces derniers phénomènes, véritables éboulements de terrain à la surface de la marge chevauchante, se produisent lorsque le pendage de celle-ci est trop important localement. La conjonction de ces trois "moteurs" doit alors entraîner un certain nombre d'effets "secondaires" comme nous l'avons déjà noté.

Tous les phénomènes répertoriés n'ont pas la même ampleur[6], aussi avons nous décidé, arbitrairement, de placer l'ensemble des phénomènes répertoriés dans trois catégories ou échelles de temps distinctes : rapide, moyenne, lente. Ainsi les phénomènes "instantanés" (à l'échelle géologique, c'est-à-dire relativement aux autres phénomènes qui s'étalent sur une période beaucoup plus longue), comme les avalanches superficielles, se produisent sur une échelle de temps dite *rapide*. La subduction de la marge océanique ainsi que l'érosion sous-crustale qui lui est corrélée, sont des phénomènes de plus grande ampleur qui se produisent sur une échelle de temps dite *lente*. Enfin, nous avons introduit une échelle de temps dite *moyenne* pour représenter les phénomènes physiques s'intercalant entre les deux classes de phénomènes précités, comme le nivellement de la fosse océanique, les affaissements de ter-

5. Ou plus exactement, pour évaluer le degré de friabilité des couches de la plaque continentale représentées par la cellule et leur potentialité à se décrocher de la plaque supérieure pour être charriées vers les profondeurs.

6. Nous voulons signifier ici qu'entre deux pas de temps donnés, le rendu-visuel engendré par chacun de ces phénomènes n'a pas la même ampleur...

rain au sein de la marge continentale et la translation générale du front de la marge océanique[7].

Pour représenter ces trois échelles de temps, nous allons imbriquer trois réseaux d'automates cellulaires distincts. Ainsi, entre deux pas de temps de l'automate à fonction de transition locale lente, il se produira autant d'itérations de l'automate à fonction de transition locale moyenne que nécessaire. Et, de la même façon, entre deux pas de temps de l'automate à fonction de transition locale moyenne, il se produira autant d'itérations de l'automate à fonction de transition locale rapide que nécessaire.

Par ailleurs, nous insistons sur les points suivants :

- en ce qui concerne les problèmes aux limites, il faut gérer séparément les cellules situées aux deux extrémités du réseau d'automates cellulaires uni-dimensionnel (la première permet, *grosso-modo* "d'introduire" de la matière dans le système alors que la dernière permet de l'éliminer),

- entre la fonction de transition globale lente et la fonction de transition globale moyenne du système dynamique, il y a transmission d'une donnée globale que nous appellerons $Coeff_{Translation}$ dans la suite du document. Ce coefficient, qui n'est pas transmis à la fonction de transition globale "rapide", permet de déclencher la translation générale du front de la marge océanique ;

- l'expression de la fonction de transition locale "rapide" est très fortement inspirée du Sand Pile Model.

D'une manière générale, d'ailleurs, ce modèle est calqué sur le principe du Sand Pile Model. Il est donc plus qualitatif que quantitatif.

5.3 Réseau d'automates cellulaires fini généralisé

Dans toute cette section, nous nous plaçons dans le cadre des réseaux d'automates cellulaires finis (c'est-à-dire ayant un nombre fini de cellules). Le but de cette section est de montrer que nous aurions très bien pu rester dans le strict cadre des réseaux d'automates cellulaires et nous passer de la donnée globale $Coeff_{Translation}$. Nous ne l'avons pas fait, comme cela est expliqué par la suite, pour des raisons de temps de calcul et de simplification à la mise en œuvre.

7. Placer cette translation de front dans la catégorie échelle de temps "moyenne" est plus osé, mais, dans la conception même de notre programme, elle n'est pas systématique puisqu'elle est subordonnée à un certain nombre de conditions supplémentaires. En fait nous souhaitions juste la dé-corréler du phénomène de plongeon de la plaque inférieure lui-même.

5.3.1 Définitions et conventions d'écriture

Soit (G, E^S, V, f) un réseau d'automates cellulaires à graphe simple, fini, non-orienté, connexe, 3-régulier où les cellules de S sont placées chacune en un point de $\{0, \ldots, n-1\}$ $(n \in \mathbb{N})$. Pour l'implémentation de ce réseau d'automates cellulaires, adoptons une structure de données de type "tableau uni-dimensionnel de cellules", une cellule étant elle-même une structure à plusieurs champs de types quelconques (les "record" du Pascal ou les "struct" du C font, par exemple, très bien l'affaire).

Soit c^0 la configuration initiale du réseau d'automates cellulaires et c^t sa configuration à la date t. Nous noterons $c^t[j]$ l'état de la j-ième cellule à la date t. Par convention $c^t[j].unChamp$ désigne la valeur du champ de nom "unChamp" de la j-ième cellule à la date t. Cette convention permet donc d'accéder sélectivement à l'ensemble des informations que comporte une cellule du réseau d'automates cellulaires.

Par la suite, nous construirons de nouveaux automates dérivant de celui-ci par simple enrichissement de l'ensemble des états d'une cellule (c'est-à-dire par ajout de champs).

Par ailleurs, l'expression $c^{t+1}[j].f(c^t[j-1], c^t[j], c^t[j+1])$, empruntée à la programmation objet, désigne le fait de mettre à jour l'état de la cellule $c^{t+1}[j]$ par application de la fonction de transition f aux données des états $c^t[j-1]$, $c^t[j]$ et $c^t[j+1]$.

5.3.2 Enoncé de la problématique

Jusqu'alors, nous n'avons considéré que des problèmes classiques du type du Sand Pile Model où l'évolution globale ne dépend que des interactions locales. Ainsi, dans l'exemple suivant d'un réseau d'automates cellulaires uni-dimensionnel fini, aucune cellule ne fait "remonter" une information locale en vue de la diffuser à l'ensemble des autres cellules.

Automates cellulaires uni-dimensionnels finis : algorithme classique

Soit $(c^t)_{t\in\mathbb{N}}$ une suite de tableaux de n cellules $(n \in \mathbb{N})$.
Initialiser c^0
Initialiser t à 0
Itérer le bloc d'instructions suivant :
   Gérer les problèmes aux limites (cas de $c^{t+1}[0]$ et $c^{t+1}[n-1]$)
   Pour j $\leftarrow$ 1 à $n-2$ faire $c^{t+1}[j].f(c^t[j-1], c^t[j], c^t[j+1])$
   Incrémenter la valeur de t d'une unité.
Jusque condition réalisée.

Envisageons à présent un problème un peu différent où l'ensemble des cellules du réseau d'automates cellulaires attendrait un signal de l'une d'entre-elles pour entamer, de façon synchrone, une nouvelle évolution.

Soit, donc, le problème suivant à résoudre : comment s'assurer de transmettre globalement (à toutes les cellules du réseau d'automates cellulaires) et "simultanément" une information issue d'une cellule particulière du réseau ?

Il s'agit en fait de préciser ici la notion de réseau d'automates cellulaires doté un mécanisme additionnel pour les communications globales.

5.3.3 Automates cellulaires finis à mécanisme additionnel pour des communications globales

Le but est donc d'écrire un réseau d'automates cellulaires tel que, entre deux pas de temps consécutifs donnés, l'ensemble des cellules puisse prendre connaissance de la valeur d'une seule et même variable globale. La valeur de cette variable globale doit pouvoir être modifiée par au moins une cellule. Pour simplifier l'exposé, supposons que cette variable globale soit un "drapeau", c'est-à-dire un booléen à valeur binaire 0 (pour faux) ou 1 (pour vrai). L'opérateur d'addition désignera la disjonction logique.

Il faut donc que, si une cellule "active" ce drapeau (fasse passer sa valeur de 0 à 1) à la date t, alors à la date $t+1$, toutes les cellules en aient connaissance. Pour cela, nous allons utiliser une suite $(drapeau^t)_{t\in\mathbb{N}}$ de booléens que nous commencerons par initialiser à 0. Les éléments de cette suite seront mis à jour au cours du temps par chacune des cellules et chacune d'elles prendra connaissance de la valeur résultante du drapeau en fin de période précédente.

Par ailleurs, il faut définir une nouvelle fonction de transition locale "étendue" $f_{étendue}$ qui, en plus du voisinage de la cellule courante, aura en argument la valeur du drapeau au pas de temps précédent. Alors, nous obtenons le nouvel algorithme suivant :

Automates cellulaires uni-dimensionnels finis dotés d'un mécanisme additionnel pour les communications globales

Soit $(c^t)_{t\in\mathbb{N}}$ une suite de tableaux de n cellules ($n \in \mathbb{N}$).
Soit $(drapeau^t)_{t\in\mathbb{N}}$ une suite de booléens.
Initialiser c^0
Initialiser toutes les valeurs de la suite $(drapeau^t)_{t\in\mathbb{N}}$ à 0.
Initialiser t à 0
Itérer le bloc d'instructions suivant :
  Gérer les problèmes aux limites (cas de $c^{t+1}[0]$ et $c^{t+1}[n-1]$)
  Pour j ← 1 à $n-2$ faire
    $drapeau^{t+1} = drapeau^{t+1} + c^{t+1}[j].f_{étendue}(c^t[j-1], c^t[j], c^t[j+1], drapeau^t)$
  Incrémenter la valeur de t d'une unité.
Jusque condition réalisée.

Pour cette solution, pouvons-nous encore raisonnablement parler de réseau d'automates cellulaires, puisque le mécanisme additionnel de communication globale ne semble guère satisfaire l'exigence de "localité" de la fonction de transition ?

5.3.4 Le mécanisme additionnel de communication globale comme cas particulier de réseau d'automates cellulaires

Considérons à présent un nouveau réseau d'automates cellulaires formé à partir du précédent par enrichissement de l'ensemble des informations stockées au niveau des cellules. Ajoutons donc un champ "drapeau" à chacune des cellules précédemment utilisées. Nous obtenons donc un nouveau réseau. Notons γ^t sa configuration à la date t. $\gamma^t[j].drapeau$, qui correspond à la valeur du drapeau de la j-ième cellule de l'automate à la date t, peut prendre la valeur 0 ou la valeur 1.

Pour diffuser à l'ensemble du réseau d'automates cellulaires une information issue d'une, quelconque, des cellules, avec des lois locales, tout en gardant le même voisinage que précédemment, nous procédons de la manière suivante :

Automates cellulaires finis de diffusion de la valeur d'un champ

```
Soit (γ^t)_{t∈ℕ} une suite de tableaux de n cellules (n ∈ ℕ).
Initialiser γ^0
Initialiser t à 0
Itérer le bloc d'instructions suivant:
   Gérer les problèmes aux limites (cas de γ^{t+1}[0] et γ^{t+1}[n-1])
   Pour j ← 1 à n-2 faire
      γ^{t+1}[j].F_{diffusion}(γ^t[j-1], γ^t[j], γ^t[j+1])
   Incrémenter la valeur de t d'une unité.
Jusque condition réalisée.
```

Où $\mathcal{F}_{diffusion}$ est une fonction de transition locale intrinsèque qui, appliquée à une cellule donnée, affecte au champ "drapeau" de cette cellule la valeur booléenne résultant d'un "ou" logique non exclusif calculé à partir des valeurs des champs "drapeaux" des trois cellules passées en argument. En clair :

$$\gamma^{t+1}[j].\mathcal{F}_{diffusion}(\gamma^t[j-1], \gamma^t[j], \gamma^t[j+1])$$

se contente d'affecter à $\gamma^{t+1}[j].drapeau$ la valeur de :

$$\gamma^t[j-1].drapeau + \gamma^t[j].drapeau + \gamma^t[j+1].drapeau$$

Il n'y a pas d'autre opération de faite au sein de la fonction de transition locale de diffusion.

Remarques : il faut appeler cette fonction de transition locale autant de fois qu'il y a de cellules (moins une) afin de s'assurer de bien toutes les informer. Ceci est, bien évidemment, très coûteux en nombre d'itérations, mais il faut bien se souvenir que l'on ignore l'endroit d'où l'information (le "signal") peut remonter... comme d'une extrémité par exemple.

5.3.5 Automates cellulaires finis dotés d'un mécanisme additionnel pour les communications globales : représentation par une imbrication d'automates cellulaires

Comme nous l'avons vu en 5.3.4, il est possible de diffuser une information locale avec une technique de réseau d'automates cellulaires. Le problème vient alors du temps mis à diffuser cette information, puisqu'il faut autant de pas de temps pour ce faire, qu'il y a de cellules (moins une), c'est-à-dire approximativement n itérations. Pour contourner cette difficulté, nous décidons d'introduire une nouvelle échelle de temps.

Ainsi, à chaque pas de temps d'application de la fonction de transition locale f (qui correspond à notre automate initial) à l'ensemble des cellules, nous effectuerons n itérations d'application de la fonction de transition locale de diffusion : $\mathcal{F}_{diffusion}$.

Imbrication de deux réseaux d'automates cellulaires pour représenter le mécanisme de diffusion globale

Soit $(\gamma^t)_{t\in\mathbb{N}}$ une suite de tableaux de n cellules ($n \in \mathbb{N}$).
Initialiser γ^0
Initialiser t à 0
Itérer le bloc d'instructions suivant :
  Gérer les problèmes aux limites (cas de $\gamma^{t+1}[0]$ et $\gamma^{t+1}[n-1]$)
  Pour j $\leftarrow$ 1 à $n-2$ faire
    $\gamma^{t+1}[j].f(\gamma^t[j-1], \gamma^t[j], \gamma^t[j+1])$
  Initialiser t' à 1
  Itérer le bloc d'instructions suivant :
    Gérer les problèmes aux limites (cas de $\gamma^{t+t'+1}[0]$ et $\gamma^{t+t'+1}[n-1]$)
    Pour j $\leftarrow$ 1 à $n-2$ faire
      $\gamma^{t+t'+1}[j].\mathcal{F}_{diffusion}(\gamma^{t+t'}[j-1], \gamma^{t+t'}[j], \gamma^{t+t'}[j+1])$
    Incrémenter la valeur de t' d'une unité.
  Jusque $t' < n$.
  Incrémenter la valeur de t de n unités.
Jusque condition réalisée.

Nous appellerons *réseau d'automates cellulaires fini généralisé* cette imbrication d'automates cellulaires finis.

5.3.6 Automates cellulaires finis dotés d'un mécanisme additionnel pour les communications globales comme cas particuliers d'automate cellulaire

Considérons à présent un nouveau réseau d'automates cellulaires formé à partir du précédent par enrichissement de l'ensemble des informations stockées au niveau des cellules. Ajoutons donc un champ "nombreDePassages" à chacune des cellules précédemment utilisées. Nous obtenons donc un nouveau réseau. Notons ξ^t sa configuration à la date t. $\xi^t[j].nombreDePassages$, qui correspond à la valeur du nombre de fois où la fonction de transition locale a été appliquée à la j-ième cellule de l'automate (à la date t), est une valeur entière.

L'algorithme précédent peut alors être réécrit de la façon suivante :

Représentation d'un réseau d'automates cellulaires fini généralisé par un réseau d'automates cellulaires à fonction de transition locale généralisée

Soit $(\xi^t)_{t\in\mathbb{N}}$ une suite de tableaux de n cellules ($n \in \mathbb{N}$).
Initialiser ξ^0
Initialiser t à 0
Itérer le bloc d'instructions suivant :
  Gérer les problèmes aux limites (cas de $\xi^{t+1}[0]$ et $\xi^{t+1}[n-1]$)
  Pour j $\leftarrow$ 1 à $n-2$ faire
    $\xi^{t+1}[j].f_{généralisée}(\xi^t[j-1], \xi^t[j], \xi^t[j+1])$
  Incrémenter la valeur de t d'une unité.
Jusque condition réalisée.

Où $f_{généralisée}$ est la fonction de transition locale suivante :

Expression de $f_{généralisée}$

Si le reste dans la division entière de $\xi^t[j].nombreDePassages$ par n vaut 1 alors :
  renvoyer $\xi^{t+1}[j].f(\xi^t[j-1], \xi^t[j], \xi^t[j+1])$
sinon :
  renvoyer $\xi^{t+1}[j].\mathcal{F}_{diffusion}(\xi^t[j-1], \xi^t[j], \xi^t[j+1])$
incrémenter l'attribut *nombreDePassages* de la cellule courante d'une unité.

5.3.7 Conclusion

Nous avons donc écrit un réseau d'automates cellulaires fini doté d'un mécanisme additionnel pour les communications globales sous forme d'une imbrication d'automates cellulaires finis (c'est-à-dire sous forme d'un réseau d'automates cellulaires fini généralisé) puis, enfin, sous forme d'un réseau d'automates cellulaires. Il a suffi, pour y parvenir, d'ajouter deux informations supplémentaires à chacune des cellules (enrichissement de l'ensemble des états) et de compliquer la fonction de transition locale donnée initialement.

5.4 Algorithme simplifié

Dans l'algorithme qui suit, nous développons une stratégie du type "jeu de bascule" entre deux tableaux de cellules (`cc` désigne le tableau courant et `c` le tableau après application de la loi de transition globale). Ce jeu de bascule se retrouve en fait imbriqué sur trois niveaux qui correspondent aux transitions lente, moyenne et rapide.

Algorithme séquentiel simplifié

```
Soient c et cc deux tableaux de NBC cellules (NBC ∈ ℕ).
Initialiser c.
Itérer le bloc d'instructions suivant :
   Recopier le contenu du tableau c dans le tableau cc
   Gérer les problèmes aux limites (cas de c[0] et c[NBC − 1])
   Pour i ← 1 à NBC − 2, faire :
      Coeff_Translation = Coeff_Translation + c[i].f_lente(cc[i − 1], cc[i], cc[i + 1])
   Itérer le bloc d'instructions suivant :
      Recopier le contenu du tableau c dans le tableau cc
      Gérer les problèmes aux limites (cas de c[0] et c[NBC − 1])
      Pour i ← 1 à NBC − 2, faire :
         c[i].f_moyenne(cc[i − 1], cc[i], cc[i + 1], Coeff_Translation)
      Itérer le bloc d'instructions suivant :
         Recopier le contenu du tableau c dans le tableau cc
         Gérer les problèmes aux limites (cas de c[0] et c[NBC − 1])
         Pour i ← 1 à NBC − 2, faire :
            c[i].f_rapide(cc[i − 1], cc[i], cc[i + 1])
      Jusque condition rapide réalisée
   Jusque condition moyenne réalisée
Si la condition de sauvegarde est vérifiée alors sauver le cliché courant.
Jusque condition lente réalisée.
```

Dans la pratique, il n'y a pas exactement "recopie du contenu du tableau c dans le tableau cc" comme il est indiqué dans l'algorithme ci-dessus. En effet, c'est un procédé beaucoup trop coûteux en terme de temps de calcul. Nous préférons utiliser une méthode de permutation des adresses des tableaux respectifs qui se résume à deux simples affectations de pointeurs.

En ce qui concerne les fonctions de transition elles-mêmes, nous les exposons ci-après en utilisant les conventions suivantes : la cellule courante est notée respectivement `CO` quand elle désigne l'ancienne valeur et `cell` quand elle désigne la nouvelle valeur à calculer. La cellule dont l'indice est inférieur d'une unité à celui de la cellule courante est notée `O` et celle dont l'indice est supérieur d'une unité à celui de la cellule courante est notée `E` (en référence aux point cardinaux).

On note respectivement de `H0` à `H6` les attributs correspondants, dans l'ordre croissant, aux hauteurs des diverses couches constitutives de la cellule courante (`H0` correspond à la hauteur de la plaque océanique en subduction et `H6` correspond à la hauteur de la couche supérieure de la plaque continentale). Les deux autres attributs de chacune des cellules sont notés `marche` et `vieil` (ils correspondent aux attributs définis en 5.2.1). Le coefficient $Coeff_{Translation}$ défini en 5.2.2 est noté `CoeffTranslation`. On note `Lmarche` et `Hmarche` la longueur et la hauteur (constantes) de toutes les "brisures" qui se forment à la surface de la marge en subduction. Enfin, `AltFosse` est une constante correspondant à l'altitude de la surface de la fosse océanique et `Delta` est un paramètre global du modèle (constant) permettant de fixer la hauteur maximale d'une "cavité" de la plaque continentale au delà de laquelle un effondrement de terrain se produit.

Fonction de transition lente

```
soient OkErosion, OkMarche et OkDegradation trois entiers initialisés à 0
# comportement par défaut
cell ← C0
si (O.H6 = 0) alors
   # cas particulier de la première cellule
   si ((C0.marche = 0) ET (alea(1000) <= Param1)) alors
      # génération d'une marche
      cell.marche ← 1
      cell.H0 ← C0.H0 + Hmarche
   sinon si (C0.marche = Lmarche) alors
      cell.marche ← 0
      cell.H0 ← C0.H0 - Hmarche
   sinon si (C0.marche >0) alors
      cell.marche ← C0.marche + 1
   sinon
      cell.marche ← 0
   fin si
   cell.H[1-6] ← cell.H0
sinon
   # mouvement de plongeon
   si (O.marche = 1) alors
      OkMarche ← Hmarche
   sinon si (C0.marche = Lmarche) alors
      OkMarche ← - Hmarche
   fin si
   # "vieillissement" de la base de la plaque continentale
   si (C0.H3 - C0.H2 >= Param2) alors
      cell.vieil ← C0.vieil
   sinon si (C0.H3 - C0.H2 >= Param3) alors
      cell.vieil ← C0.vieil + 1
   sinon si (C0.H3 - C0.H2 = 1) alors
      cell.vieil ← C0.vieil + 4
   sinon si ((C0.H3 = C0.H2) ET (sousPlaqueContinentale(C0) = vrai)) alors
      cell.vieil ← C0.vieil + 10
   sinon
      cell.vieil ← 0
   fin si
   # dégradation de la marge continentale et re-mise a jour de "vieil"
   si (sousPlaqueContinentale(C0) = vrai) alors
      si (((C0.H6 >C0.H3) ET (C0.H3 >= C0.H4) ET
         (C0.H3 >= C0.H5) ET (cell.vieil >Param4 + alea(Param5))) OU
         ((C0.H5 >C0.H3) ET (C0.H3 >= C0.H4) ET (cell.vieil >Param6)) OU
         ((C0.H4 >C0.H3) ET (cell.vieil >Param7))) alors
         cell.vieil ← 0
         OkDegradation ← 1
      fin si
   fin si
   # Transmission de la marche
   cell.marche ← O.marche
```

```
   # érosion de la base de la plaque continentale
   si ((C0.H3 >C0.H2) ET (sousPlaqueContinentale(C0) = vrai)) alors
      si ((sousPlaqueContinentale(0) = faux) ET (alea(2) = 1)) alors
         # érosion du front de la plaque continentale avec une probabilité de 50%
         OkErosion ← 1
      sinon si ((alea(1000) <= 10) ET (sousPlaqueContinentale(0) = vrai) ET
         (C0.H3 >C0.H2+1)) alors
         # érosion sous-crustale avec une probabilité de 1%
         OkErosion ← 1
      fin si
   fin si
   # Avancée du front de la subduction
   si ((sousPlaqueContinentale(0) = faux) ET (sousPlaqueContinentale(C0) = vrai) ET
      (OkErosion = 1) ET (C0.H3 = C0.H2 + 1) ET (C0.H3 >= C0.H4) ET
      (C0.H3 >= C0.H5) ET (C0.H3 >= C0.H6)) alors
      CoeffTranslation ← 1
   fin si
   cell.H0 ← C0.H0 + OkMarche
   cell.H1 ← C0.H0 + OKMarche + (0.H1 - 0.H0) + OkErosion
   # cas des cellules situees avant la fosse
   si (avantFosse(C0) = vrai) alors
      # cas des cellules situees avant la fosse
      cell.H[2-6] ← cell.H1
   sinon
      # cas des autres cellules
      cell.H2 ← MAX((C0.H2 + OkErosion) , cell.H1)
      cell.H3 ← MAX((C0.H3 - CoeffTranslation + OkDegradation) , cell.H2)
      cell.H4 ← MAX((C0.H4 - CoeffTranslation) , cell.H3)
      cell.H5 ← MAX((C0.H5 - CoeffTranslation) , cell.H4)
      cell.H6 ← MAX((C0.H6 - CoeffTranslation) , cell.H5)
   fin si
fin si
```

Fonction de transition moyenne

```
# comportement par défaut
cell ← C0
si (0.H6 = 0) alors
   # cas particulier de la première cellule
   si (CoeffTranslation = 1) alors
      si (C0.marche = Lmarche) alors
         cell.marche ← 0
         cell.H0 ← C0.H0 - Hmarche
      sinon si (C0.marche >0) alors
         cell.marche ← C0.marche + 1
      sinon
         cell.marche ← 0
      fin si
      cell.H[1-6] ← cell.H0
   fin si
sinon
   # cas des autres cellules
   si (CoeffTranslation = 1) alors
```

```
      cell.H0 ← O.H0
      cell.H1 ← O.H1
      cell.marche ← O.marche
   fin si
   si (sousPlaqueContinentale(CO) = faux) alors
      si (CoeffTranslation = 1) alors
         cell.H[2-6] ← MAX(O.H1,AltFosse)
      sinon
         cell.H[2-6] ← MAX(CO.H1,AltFosse)
      fin si
   sinon
      # on a donc (sousPlaqueContinentale(CO) = vrai)
      si (((sousPlaqueContinentale(E) = vrai) ET (CO.H2 >E.H2 + Delta)) OU
         ((sousPlaqueContinentale(O) = vrai) ET (CO.H2 >O.H2 + Delta))) alors
         # affaissement de terrain au sein de la plaque continentale
         cell.H2 ← CO.H2 - Delta
         cell.H3 ← CO.H3 - Delta
         cell.H4 ← CO.H4 - Delta
         cell.H5 ← CO.H5 - Delta
         cell.H6 ← CO.H6 - Delta
      fin si
   fin si
fin si
```

Fonction de transition rapide

```
soient cTOgd, dTOc, gTOc trois entiers initialisés à 0
# comportement par défaut
cell ← CO
si (CO.H6 - CO.H2 >= 3) alors
   # on peut retirer deux blocs a la colonne courante
   si (CO.H6 >= O.H6 + 2) alors cTOgd ← cTOgd + 1
   si (CO.H6 >= E.H6 + 2) alors cTOgd ← cTOgd + 1
fin si
si ((E.H6 >= CO.H6 + 2) ET (E.H6 - E.H2 >= 3)) alors
   # on peut retirer un bloc a la colonne de droite
   dTOc ← dTOc + 1
fin si
si ((O.H6 >= CO.H6 + 2) ET (O.H6 - O.H2 >= 3)) alors
   # on peut retirer un bloc a la colonne de gauche
   gTOc ← gTOc + 1
fin si
cell.H6 ← CO.H6 - cTOgd + dTOc + gTOc
cell.H5 ← MIN(cell.H6,CO.H5)
cell.H4 ← MIN(cell.H6,CO.H4)
cell.H3 ← MIN(cell.H6,CO.H3)
```

Chapitre 6

Le modèle bi-dimensionnel

Tout comme dans le cas du modèle uni-dimensionnel, nous nous sommes fortement inspirés ici encore des principes du Sand Pile Model. Pour commencer, nous avons donc cherché à implémenter un modèle d'avalanches dans un tas de sable représenté par un réseau d'automates cellulaires bi-dimensionnel. Constatant que la multiplication des informations stockées dans la structure même du réseau nous offre un meilleur rendu-visuel, nous avons alors cherché à généraliser cette approche au phénomène de subduction-érosion. Au cours de ce chapitre, après avoir présenté un modèle du tas de sable bi-dimensionnel, nous exposerons donc les principes qui nous ont amenés à le généraliser puis nous aborderons la description de notre propre réseau d'automates cellulaires en commençant par décrire le "pavage du plan", l'ensemble des états d'une cellule, son voisinage, la fonction de transition locale et la configuration initiale. Nous présenterons aussi l'algorithme simplifié correspondant.

6.1 Le tas de sable bi-dimensionnel

Contrairement au réseau d'automates cellulaires uni-dimensionnel où chaque grain de sable n'a pas vraiment "d'existence propre" puisque l'entité de base est la colonne de grains, nous cherchons ici à individualiser les grains du tas de sable par coloration. A chaque cellule du réseau, donc, nous proposons d'associer trois informations de type entier naturel borné correspondant respectivement à la nature du matériel composant la cellule, sa couleur (il peut-y avoir plusieurs couleurs imposées artificiellement à une même cellule pour faciliter le repérage ou en fonction d'autres critères...) et sa position par rapport à une ligne de niveau de référence (cet attribut peut prendre les valeurs -1, 0 ou +1, nous aurions pu nous en passer pour ce modèle-ci mais, il nous a paru important de l'introduire dès à présent même s'il ne trouve sa justification que pour le réseau d'automates cellulaires modélisant le phénomène de subduction-érosion).

Pour ce réseau, nous avons écrit une règle de transition déterministe telle que, à chaque pas de temps et simultanément avec l'ensemble des autres cellules du tas, une cellule adopte ou non une nouvelle "matière" et une nouvelle "couleur" en fonction des cellules composant son voisinage. Pour identifier précisément chaque cellule voisine, nous la référençons relativement à la cellule courante à l'aide des points cardinaux. Ainsi, la cellule d'identifiant `NNE` correspond à la cellule dont l'indice de ligne est celui de la cellule courante incrémenté de deux unités et dont l'indice de colonne est celui de la cellule courante incrémenté d'une unité. La cellule courante est notée respectivement `CO` quand elle désigne l'ancienne valeur et `cell` quand elle désigne la nouvelle valeur à calculer.

Pour faciliter la lecture de l'algorithme de la loi de transition locale, nous proposons la grille suivante (fig. 6.1) qui permet d'identifier toute cellule voisine relativement à une cellule courante donnée.

NNOO	**NNO**	**NN**	**NNE**	**NNEE**
OON	**NO**	**N**	**NE**	**EEN**
OO	**O**	**CO**	**E**	**EE**
OOS	**SO**	**S**	**SE**	**EES**
SSOO	**SSO**	**SS**	**SSE**	**SSEE**

FIG. 6.1 – *Identification des cellules voisines d'une cellule courante donnée à base de "points cardinaux"*

Automates cellulaires du SPM bi-dimensionnel : algorithme

```
#############################
# Loi de transition locale
#############################
# comportement par défaut
cell ← CO
# autres comportements
si ((SS.altitude ≥ 0) ET (CO.nature ≠ NAT_BORD)) alors
   #############################
   # avalanche possible
   #############################
   si ((CO.nature = NAT_SOCLE_CONTI) OU (CO.nature = NAT_SOCLE_CONTI_ERO)) alors
      #############################
      # processus de soustraction
      #############################
      si ((S.nature = NAT_EAU) OU (S.nature = NAT_COUCHE_INTERSTI)) alors
```

```
        # chute libre
        cell ← (S.couleur,S.nature,CO.altitude)
    sinon si (((N.nature = NAT_EAU) OU (N.nature = NAT_COUCHE_INTERSTI)) ET
        ((((NO.nature = NAT_EAU) OU (NO.nature = NAT_COUCHE_INTERSTI)) ET
        ((O.nature = NAT_EAU) OU (O.nature = NAT_COUCHE_INTERSTI)) ET
        ((SO.nature = NAT_EAU) OU (SO.nature = NAT_COUCHE_INTERSTI))) OU
        (((NE.nature = NAT_EAU) OU (NE.nature = NAT_COUCHE_INTERSTI)) ET
        ((E.nature = NAT_EAU) OU (E.nature = NAT_COUCHE_INTERSTI)) ET
        ((SE.nature = NAT_EAU) OU (SE.nature = NAT_COUCHE_INTERSTI)))) ET
        ((SS.nature = NAT_SOCLE_CONTI) OU (SS.nature = NAT_SOCLE_CONTI_ERO))) alors
        # avalanche bloc haut de pile
        cell ← (CLR_EAU,NAT_EAU,CO.altitude)
    sinon si (((NN.nature = NAT_EAU) OU (NN.nature = NAT_COUCHE_INTERSTI)) ET
        ((N.nature = NAT_SOCLE_CONTI) OU (N.nature = NAT_SOCLE_CONTI_ERO)) ET
        ((NNO.nature = NAT_EAU) OU (NNO.nature = NAT_COUCHE_INTERSTI)) ET
        ((NO.nature = NAT_EAU) OU (NO.nature = NAT_COUCHE_INTERSTI)) ET
        ((O.nature = NAT_EAU) OU (O.nature = NAT_COUCHE_INTERSTI)) ET
        ((NNE.nature = NAT_EAU) OU (NNE.nature = NAT_COUCHE_INTERSTI)) ET
        ((NE.nature = NAT_EAU) OU (NE.nature = NAT_COUCHE_INTERSTI)) ET
        ((E.nature = NAT_EAU) OU (E.nature = NAT_COUCHE_INTERSTI))) alors
        # avalanche bloc sous haut de pile
        cell ← (CLR_EAU,NAT_EAU,CO.altitude)
    fin si
  fin si
  si ((O.nature = NAT_SOCLE_CONTI) OU (O.nature = NAT_SOCLE_CONTI_ERO) OU
    (NO.nature = NAT_SOCLE_CONTI) OU (NO.nature = NAT_SOCLE_CONTI_ERO) OU
    (N.nature = NAT_SOCLE_CONTI) OU (N.nature = NAT_SOCLE_CONTI_ERO) OU
    (NE.nature = NAT_SOCLE_CONTI) OU (NE.nature = NAT_SOCLE_CONTI_ERO) OU
    (E.nature = NAT_SOCLE_CONTI) OU (E.nature = NAT_SOCLE_CONTI_ERO)) alors
    ############################
    # processus d'addition
    ############################
    si (((N.nature = NAT_SOCLE_CONTI) OU (N.nature = NAT_SOCLE_CONTI_ERO)) ET
        ((CO.nature = NAT_EAU) OU (CO.nature = NAT_COUCHE_INTERSTI))) alors
        # chute libre
        cell ← (N.couleur,N.nature,CO.altitude)
    sinon si (((CO.nature = NAT_EAU) OU (CO.nature = NAT_COUCHE_INTERSTI)) ET
        ((N.nature = NAT_EAU) OU (N.nature = NAT_COUCHE_INTERSTI)) ET
        ((NN.nature = NAT_EAU) OU (NN.nature = NAT_COUCHE_INTERSTI))) alors
        # avalanche
        si (((OO.nature = NAT_EAU) OU (OO.nature = NAT_COUCHE_INTERSTI)) ET
            ((OON.nature = NAT_EAU) OU (OON.nature = NAT_COUCHE_INTERSTI)) ET
            ((NNOO.nature = NAT_EAU) OU (NNOO.nature = NAT_COUCHE_INTERSTI)) ET
            ((O.nature = NAT_SOCLE_CONTI) OU (O.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NO.nature = NAT_SOCLE_CONTI) OU (NO.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NNO.nature = NAT_EAU) OU (NNO.nature = NAT_COUCHE_INTERSTI)) ET
            ((SO.nature = NAT_SOCLE_CONTI) OU (SO.nature = NAT_SOCLE_CONTI_ERO))) alors
            cell ← (O.couleur,O.nature,CO.altitude)
        sinon si ((((OO.nature = NAT_SOCLE_CONTI) OU (OO.nature = NAT_SOCLE_CONTI_ERO)) OU
            ((OON.nature = NAT_SOCLE_CONTI) OU (OON.nature = NAT_SOCLE_CONTI_ERO)) OU
            ((NNOO.nature = NAT_SOCLE_CONTI) OU (NNOO.nature = NAT_SOCLE_CONTI_ERO))) ET
            (((O.nature = NAT_SOCLE_CONTI) OU (O.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NO.nature = NAT_SOCLE_CONTI) OU (NO.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NNO.nature = NAT_EAU) OU (NNO.nature = NAT_COUCHE_INTERSTI)))) alors
```

```
            cell ← (NO.couleur,NO.nature,CO.altitude)
         sinon si ((NON (((O.nature = NAT_SOCLE_CONTI) OU (O.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NO.nature = NAT_SOCLE_CONTI) OU (NO.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NNO.nature = NAT_EAU) OU (NNO.nature = NAT_COUCHE_INTERSTI)))) ET
            ((E.nature = NAT_SOCLE_CONTI) OU (E.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NE.nature = NAT_SOCLE_CONTI) OU (NE.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NNE.nature = NAT_EAU) OU (NNE.nature = NAT_COUCHE_INTERSTI)) ET
            ((SE.nature = NAT_SOCLE_CONTI) OU (SE.nature = NAT_SOCLE_CONTI_ERO))) alors
            cell ← (NE.couleur,NE.nature,CO.altitude)
         sinon si (((SO.nature = NAT_SOCLE_CONTI) OU (SO.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((O.nature = NAT_SOCLE_CONTI) OU (O.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NO.nature = NAT_EAU) OU (NO.nature = NAT_COUCHE_INTERSTI)) ET
            ((SE.nature = NAT_SOCLE_CONTI) OU (SE.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((E.nature = NAT_SOCLE_CONTI) OU (E.nature = NAT_SOCLE_CONTI_ERO)) ET
            ((NE.nature = NAT_EAU) OU (NE.nature = NAT_COUCHE_INTERSTI)) ET
            ((S.nature = NAT_EAU) OU (S.nature = NAT_COUCHE_INTERSTI))) alors
            cell ← (E.couleur,E.nature,CO.altitude)
         fin
      fin
   fin
sinon si (S.altitude = 0) ET
   ((CO.nature = NAT_EAU) OU (CO.nature = NAT_COUCHE_INTERSTI)) ET
   ((N.nature = NAT_SOCLE_CONTI) OU (N.nature = NAT_SOCLE_CONTI_ERO)) alors
   cell ← (N.couleur,N.nature,CO.altitude)
sinon
   cell ← (CO.couleur,CO.nature,CO.altitude)
fin
```

Le système présenté sur les fig. 6.2 correspond à l'évolution d'une configuration initiale rectangulaire d'un tas de sable. L'algorithme que nous proposons peut cependant très bien s'appliquer, sans aucune adaptation, à un autre type de configuration de départ.

FIG. 6.2 – *Copies d'écran du Sand Pile Model bi-dimensionnel*

Comme nous pouvons le constater à la lecture de l'algorithme précédent, la mise en œuvre d'une simulation du Sand Pile Model est moins immédiate en deux dimensions qu'en une seule !

6.2 Le système dynamique discret

Conscient du fait que le découpage en cellules du modèle uni-dimensionnel est trop "simplificateur" (7 entiers nous suffisent pour représenter une coupe verticale de 5 km de haut sur 25 m de large, correspondant au pourtour de la fosse océanique), nous avons cherché à modéliser le phénomène géotectonique par un réseau d'automates cellulaires bi-dimensionnel. Dans ce nouveau découpage, une cellule ne représente plus désormais qu'une portion d'espace homogène de 25 m par 25 m.

6.2.1 Pavage de la zone et ensemble des états d'une cellule

Alors que dans le modèle uni-dimensionnel, nous faisons un découpage en colonnes verticales de la zone de subduction, dans le cas bi-dimensionnel, le découpage est fait par blocs.

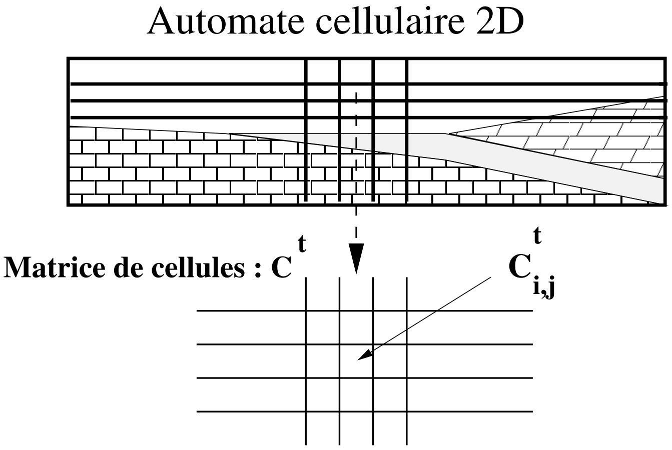

FIG. 6.3 – *Maillage bi-dimensionnel de la zone de subduction*

L'état d'une cellule est désormais déterminé par la donnée de six entiers naturels bornés appelés respectivement : `couleur`, `nature`, `vieil`, `courbure`, `memoire`, `altitude`. L'attribut de `couleur` permet de distinguer, à la visualisation, deux cellules de même `nature`. L'attribut `courbure` permet, pour une cellule donnée de la plaque océanique ou du chenal de subduction, de progresser avec la bonne inclinaison (exprimée en %). Son compteur `memoire` est incrémenté d'une unité à chaque "progression horizontale" et remis à zéro lors d'une "progression verticale" (c'est-à-dire lorsque la valeur de l'attribut `courbure` correspondant à la prochaine position de la cellule courante est atteinte). L'attribut `vieil` permet de modéliser un effet de dégradation de la cohésion de la plaque continentale avec transformation de matière. Enfin l'`altitude`, est utilisée pour faciliter l'implémentation des phénomènes d'avalanches superficielles, afin d'éviter qu'ils ne se produisent pour les cellules de la base de la marge continentale. Cet attribut est simplement destiné à améliorer la cohabitation des diverses règles de transition que nous mettons en œuvre.

Ces six entiers sont bornés, l'état d'une cellule du réseau d'automates cellulaires est donc un élément d'un ensemble fini.

6.2.2 Fonction de transition locale

Principes généraux de la fonction de transition locale

```
#############################
# Loi de transition locale
#############################
# comportement par défaut
cell ← CO
# autres comportements
si (CO.nature = NAT_BORD)
   #cas particulier d'une cellule aux bords du domaine
   cell ← CO
sinon si ((numIteration%echelle_temps) = 0) alors
   # phénomènes associés à l'échelle de temps "lente"
   cell.courbure ← CO.courbure
   cell.memoire ← ((O.memoire ≥ (CO.courbure-1))? 0: O.memoire+1)
   cell.altitude ← CO.altitude
   si (O.nature = NAT_BORD) alors
      # introduction de plaque océanique sur le bord latéral gauche
   sinon si ((CO.nature ≠ NAT_SOCLE_OCEAN) ET
      (O.nature ≠ NAT_SOCLE_OCEAN) ET
      (NO.nature ≠ NAT_SOCLE_OCEAN)) alors
      # cas d'une cellule n'appartenant pas à la plaque océanique ou
      # n'étant pas amenée à être recouverte par une cellule de plaque
      # océanique au pas de temps suivant
      si ((CO.nature = NAT_SOCLE_CONTI) ET
         ((S.nature = NAT_SOCLE_CONTI_ERO)
         OU (S.nature = NAT_COUCHE_INTERSTI))) alors
         # vieillissement de la base de la plaque continentale
         # (par incrémentation aléatoire du coefficient VIEIL) suivie ou non
         # d'une transformation chimique de la cellule (CLR_SOCLE_CONTI_ERO,
         # NAT_SOCLE_CONTI_ERO...)
      sinon si ((N.nature = NAT_EAU) ET
         (CO.altitude = 1) ET
         (CO.nature = NAT_COUCHE_INTERSTI)) alors
         # comblement de la couche supérieure de la fosse océanique
      sinon si ((CO.altitude <1) OU
         ((CO.altitude = 1) ET (CO.nature = NAT_SOCLE_CONTI_ERO))) alors
         # plongeon de la couche interstitielle et des cellules érodées
      fin si
   sinon
      # gérer le plongeon de la plaque océanique
   fin si
sinon
   # phénomènes associés à l'échelle de temps "rapide"
   si (((CO.nature = NAT_SOCLE_CONTI) OU (CO.nature = NAT_SOCLE_CONTI_ERO)) ET
      ((N.nature = NAT_SOCLE_CONTI) OU (N.nature = NAT_SOCLE_CONTI_ERO)) ET
      ((NO.nature = NAT_SOCLE_CONTI) OU (NO.nature = NAT_SOCLE_CONTI_ERO)) ET
      ((O.nature = NAT_SOCLE_CONTI) OU (O.nature = NAT_SOCLE_CONTI_ERO)) ET
```

```
    ((SO.nature = NAT_SOCLE_CONTI) OU (SO.nature = NAT_SOCLE_CONTI_ERO)) ET
    ((SSO.nature = NAT_SOCLE_CONTI) OU (SSO.nature = NAT_SOCLE_CONTI_ERO)) ET
    (E.nature ≠ NAT_BORD) ET (S.nature = NAT_COUCHE_INTERSTI)) alors
    # macro affaissement de terrain au sein de la plaque continentale
 sinon si ((CO.altitude >1) ET
    (CO.nature = NAT_SOCLE_CONTI_ERO) ET
    ((S.nature = NAT_COUCHE_INTERSTI) OU (S.nature = NAT_EAU))) alors
    # micro affaissement de socle continental erodé
 sinon si ((SS.altitude ≥ 1) ET (CO.nature ≠ NAT_BORD)) alors
    # avalanches superficielles envisageables. Reprendre et adapter
    # l'algorithme du Sand Pile Model bi-dimensionnel
 sinon si ((CO.altitude <← 1) ET
    (CO.nature = NAT_EAU) ET
    ((N.nature = NAT_COUCHE_INTERSTI) OU
    (E.nature = NAT_COUCHE_INTERSTI) OU
    (NE.nature = NAT_COUCHE_INTERSTI))) alors
    # comblement de la fosse oceanique
```

6.3 Algorithme simplifié

Tout comme dans le cas uni-dimensionnel, nous ne pouvons pas considérer un tableau bi-infini de cellules pour des raisons d'ordre technique (par ailleurs, il s'agit de modéliser une portion de coupe de zone de subduction de 25 km de long sur 5 km de haut). Notre réseau d'automates cellulaires est donc une matrice finie d'automates. En ce qui concerne les extrémités[1], nous avons choisi de les singulariser par un état "frontière" invariant et de les répliquer symétriquement aux bords de façon à obtenir l'effet d'un tore bi-dimensionnel[2]. En effet, puisque chaque cellule évolue en fonction de la donnée des états de ses 24 cellules voisines, les cellules des bords du domaine ont elles-aussi besoin de connaître l'état de leurs voisines à chaque pas de temps. Nous avons donc introduit dans le modèle un ensemble de cellules (appelées cellules du bord externe), qui n'a pas à proprement parler "d'existence réelle" vis-à-vis de la modélisation. Ces cellules étendent le réseau d'automates cellulaires et sont stockées sur le pourtour du domaine étudié. Elles ajoutent ainsi deux rangées ou colonnes au domaine et bordent les cellules qui sont situées sur les bords "intérieurs" (voir fig. 6.4).

Dans l'algorithme qui suit, i désigne l'indice de colonne et j l'indice de ligne.

Algorithme séquentiel simplifié

```
Soient c et cc deux tableaux de NBL × NBC cellules (NBL,NBC ∈ ℕ).
Initialiser c.
```

1. Qui dans notre cas pratique représentent environ 1,2% de l'ensemble des cellules.
2. Nous sommes bien conscients que cette réplication ne s'impose pas ici et que nous pouvons nous contenter de travailler modulo le nombre des cellules. Toutefois, nous présentons cette technique dans le but d'introduire celle employée pour la simulation parallèle.

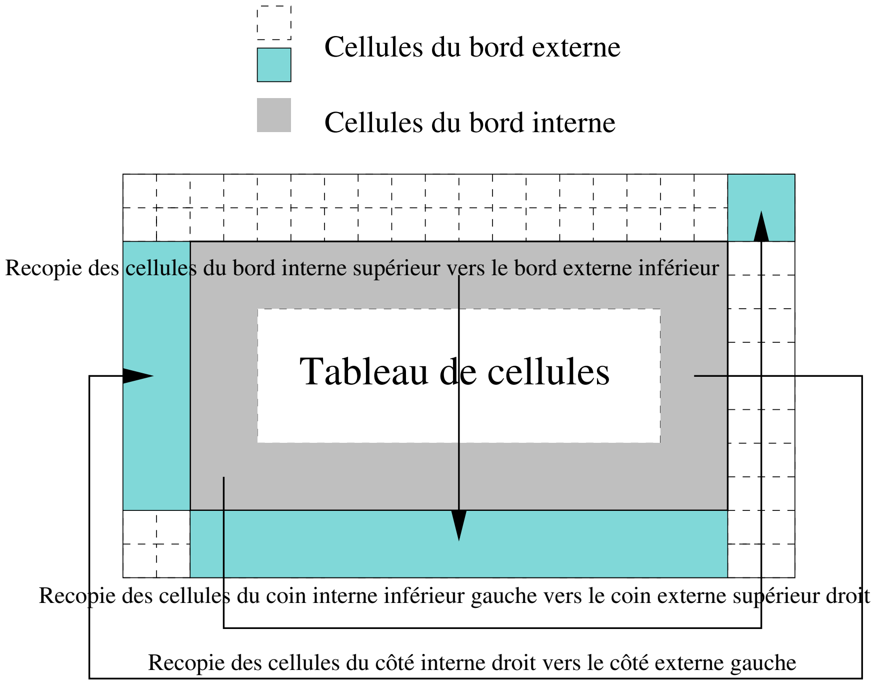

FIG. 6.4 – *Traitement des extrémités du réseau d'automates cellulaires, cas bi-dimensionnel*

`compteur_d_iteration` $\leftarrow 0$
Itérer le bloc d'instructions suivant :
  Recopier le contenu du tableau c dans le tableau cc
  # Recopie des lignes des bords "intérieurs" sur les bords "extérieurs"
  Pour $i \leftarrow 0$ à 1, faire :
    Pour $j \leftarrow 2$ à $NBL-3$, faire :
      $cc_{i+NBC-2}^{j} \leftarrow cc_{i+2}^{j}$
      $cc_{i}^{j} \leftarrow cc_{i+NBC-4}^{j}$
  # Recopie des colonnes des bords "intérieurs" sur les bords "extérieurs"
  Pour $i \leftarrow 2$ à $NBL-3$, faire :
    Pour $j \leftarrow 0$ à 1, faire :
      $cc_{i}^{j+NBL-2} \leftarrow cc_{i}^{j+2}$
      $cc_{i}^{j} \leftarrow cc_{i}^{j+NBL-4}$
  # Recopie des angles des bords "intérieurs" sur les bords "extérieurs"
  Pour $i \leftarrow 0$ à 1, faire :
    Pour $j \leftarrow 0$ à 1, faire :
      $cc_{i}^{j} \leftarrow cc_{i+NBC-4}^{j+NBL-4}$
      $cc_{i}^{j+NBL-2} \leftarrow cc_{i+NBC-4}^{j+2}$
      $cc_{i+NBC-2}^{j} \leftarrow cc_{i+2}^{j+NBL-4}$
      $cc_{i+NBC-2}^{j+NBL-2} \leftarrow cc_{i+2}^{j+2}$
  # Application de la fonction de transition à proprement parler à toutes
  # les cellules du domaine exceptées celles des bords "extérieurs"
  Pour $i \leftarrow 2$ à $NBC-3$, faire :
    Pour $j \leftarrow 2$ à $NBL-3$, faire :

$$
\begin{aligned}
c_i^j \leftarrow f(&cc_{i+2}^{j-2}, cc_{i+2}^{j-1}, cc_{i+2}^{j}, cc_{i+2}^{j+1}, cc_{i+2}^{j+2},\\
&cc_{i+1}^{j-2}, cc_{i+1}^{j-1}, cc_{i+1}^{j}, cc_{i+1}^{j+1}, cc_{i+1}^{j+2},\\
&cc_{i}^{j-2}, cc_{i}^{j-1}, cc_{i}^{j}, cc_{i}^{j+1}, cc_{i}^{j+2},\\
&cc_{i-1}^{j-2}, cc_{i-1}^{j-1}, cc_{i-1}^{j}, cc_{i-1}^{j+1}, cc_{i-1}^{j+2},\\
&cc_{i-2}^{j-2}, cc_{i-2}^{j-1}, cc_{i-2}^{j}, cc_{i-2}^{j+1}, cc_{i-2}^{j+2},\\
&\texttt{compteur\_d\_iteration})
\end{aligned}
$$

```
   compteur_d_iteration ← compteur_d_iteration +1
Tant que condition de fin de boucle non réalisée
```

Dans la pratique, il n'y a pas exactement "recopie du contenu du tableau c dans le tableau cc" comme il est indiqué dans l'algorithme ci-dessus. En effet, c'est un procédé beaucoup trop coûteux en terme de temps de calcul. Nous préférons utiliser une méthode de permutation des adresses des tableaux respectifs qui se résume à deux simples affectations de pointeurs.

Troisième partie

Les simulations et l'optimisation parallèle

Chapitre 7

Aspects matériel et logiciel du parallélisme

Ce travail, nous le verrons plus tard au niveau des comparaisons des performances des simulations séquentielles et parallèles, justifie une fois de plus l'intérêt du parallélisme dans la recherche en informatique. Les grands projets actuels, en base de données, visualisation, langages, systèmes d'exploitation ou architecture, ont tous désormais une dimension parallèle dans leur développement. Même si la puissance des machines mono-processeurs augmente elle-même continuellement, il est un fait, comme le constate [JEZEQUEL, 1996], que les besoins en puissance de calcul croissent en même temps que les performances des machines. Il s'impose donc de faire coopérer des unités de calcul toujours plus puissantes pour satisfaire les besoins applicatifs toujours plus nombreux et gourmands.

La course à la puissance a donné naissance à différents types de systèmes parallèles que l'on peut classer selon l'organisation de la mémoire, les schémas d'adressage et les mécanismes de communication entre processus. La plupart des machines parallèles du marché possédant un relativement grand nombre de processeurs sont des systèmes à mémoire distribuée ou encore des réseaux d'unités de calcul. Ces architectures ont en commun une capacité d'extensibilité* (ou scalability*) d'une ou plusieurs de leurs caractéristiques comme le nombre d'unités de calcul, la taille de la mémoire, le débit du réseau...Cette capacité d'extensibilité à coût de développement constant comme le note [LE GUYADEC, 1995], ouvre la voie à la résolution de problèmes d'une complexité telle qu'ils étaient jusqu'alors considérés comme hors d'atteinte. La difficulté est de parvenir à exploiter au mieux la puissance des machines parallèles.

L'objet de ce chapitre est de présenter les caractéristiques fondamentales des machines parallèles, les modèles de programmation parallèle et un ensemble de propriétés relatives au parallélisme.

7.1 Généralités sur les architectures et les réseaux

Une machine parallèle se définit comme une **collection d'éléments de calcul** capables de **communiquer** et **coopérer** dans le but de résoudre **rapidement** des problèmes de **grande taille** [Lalevée, 1995, Desprez, 1996a]. Cette simple assertion introduit les divers points que nous allons aborder au cours de ce chapitre. Commençons par les expliciter.

7.1.1 Les constats

Collection d'éléments de calcul : Le premier problème posé est d'identifier le nombre d'éléments de calcul à mettre en place et la puissance de chacun de ces éléments. Il faut aussi évaluer la taille de mémoire qui doit leur être associée, et envisager le type d'entrées-sorties à réaliser. Enfin, tout ce qui précède ayant été satisfait, il faut les disposer les uns par rapport aux autres.

Communiquer : Ces éléments ayant été disposés, il s'agit alors de leur permettre d'échanger des messages. Par exemple, faire communiquer p unités de calcul nécessite de prévoir C_p^2 canaux de communication. Il faut donc envisager d'adopter un réseau d'interconnexion adéquat au moindre coût. Il faut enfin envisager un protocole d'échange d'informations.

Coopérer : Un autre problème soulevé est celui qui consiste à faire coopérer toutes ces unités pour réaliser un travail donné ; cela nécessite souvent de mettre en place des mécanismes de synchronisation.

Rapidement : L'augmentation des vitesses d'exécution des algorithmes conduit à chercher des mesures fiables des performances atteintes afin de se livrer à des comparaisons fondées.

Grande taille : Le dernier problème est parfois celui du gigantisme des problèmes à résoudre. Gigantisme par la taille des données à stocker et à communiquer, mais aussi gigantisme par le temps de calcul à y consacrer. Il faut alors envisager une algorithmique adaptée : modèle de calcul approprié, taux de parallélisme potentiel de l'application, degré de spécialisation d'une architecture au problème fixé, type même de parallélisme à employer (implicite ou explicite)...

Pour répondre à ces interrogations, il est d'usage de présenter la classification des machines parallèles de Flynn.

7.1.2 Classifications

Taxinomie de Flynn [Flynn, **1966**, Flynn, **1972**]

Cette classification est fondée sur la notion de flots d'information qui transitent sous deux types dans un processeur : instructions et données. Elle divise l'ensemble du parc des machines parallèles en quatre catégories selon que leurs processeurs traitent simultanément un ou plusieurs flots de chacun de ces deux types. On distingue ainsi :

- les machines SISD* (Single Instruction stream, Single Data stream) : ces machines correspondent aux machines séquentielles classiques (machine de Von Neumann). L'amélioration de leurs performances passe obligatoirement par l'accélération de leurs cycles d'intructions et de leurs temps d'accès à la mémoire. Les lois physiques (vitesse de commutation des bascules, capacité d'intégration...) apportent une limite théorique à leurs évolutions ;
- les machines SIMD* (Single Instruction stream, Multiple Data streams) : il s'agit d'une classe d'architectures composées d'une seule unité de contrôle et de plusieurs unités de traitement (tableau de processeurs). Tous les éléments de calcul reçoivent de l'unité de contrôle globale les instructions à exécuter et exécutent simultanément la même opération sur leurs données propres. Il s'agit de la mise en œuvre directe du parallélisme de données sur lequel nous reviendrons dans la suite de ce chapitre. L'exécution des instructions est synchrone, du point de vue du programmeur, un programme parallèle SIMD se comporte de la même manière qu'un programme séquentiel ;
- les machines MISD* (Multiple Instruction streams, Single Data stream) : dans ces architectures, le traitement est exécuté en plusieurs étapes sur la même donnée (modèle pipeline*). Chaque processeur reçoit des instructions distinctes opérant sur un unique flot de données. La sortie d'une unité de calcul devient donc l'entrée d'une autre unité de calcul. On y classe les architectures systoliques* ;
- les machines MIMD* (Multiple Instruction streams, Multiple Data streams) : il s'agit des machines multi-processeurs dans lesquelles les processeurs sont autonomes (ils disposent de leur propre unité de contrôle). L'exécution est asynchrone et les programmes exécutés par chacun des processeurs peuvent être différents.

Nous nous intéresserons plus particulièrement aux architectures de type MIMD. En effet, bien que des recherches soient toujours en cours pour l'implémentation sur machines SIMD[1], les contraintes technologiques imposées

1. Les machines CM-1, CM-2, la MasPar MP-1 et la MasPar MP-2 sont des exemples de machines SIMD...

par ce type de machines ont conduit les constructeurs à leur préférer les machines MIMD [2]. D'autre part, le modèle SIMD est plus approprié pour des applications très spécifiques et difficilement adaptable à tout type d'application comme peut l'être le modèle MIMD.

Cette classification nécessite d'être précisée par l'organisation de la mémoire elle-même.

Classification selon les accès à la mémoire

Il existe en fait deux classes principales extrêmes de machines MIMD. La différence entre les deux réside dans le partage de la mémoire et les mécanismes de communication entre processeurs :

- mémoire partagée*: dans ce cas, l'ensemble des processeurs accèdent à une mémoire centrale commune par l'intermédiaire d'un réseau d'interconnexion. Cette catégorie se scinde elle-même en deux sous catégories selon le type d'espace d'adressage: architectures de type SASM* (Single Address space Shared Memory) avec accès mémoire de type UMA* (Uniform Memory Access, le coût de l'accès aux données est le même quel que soit l'endroit où elles sont rangées) ou de type SADM* (Single Address space Distributed Memory). Dans ce dernier cas de machines MIMD à espace d'adressage de type SADM, on envisage encore quatre types d'accès mémoire bien distincts: NUMA* (Non Uniform Memory Access, où les performances sont dépendantes de la localisation des données), CC-NUMA* (Cache Coherent NUMA, qui est une variante des NUMA avec des caches), OSMA* (Operating System Memory Access, où le système d'exploitation se charge des accès mémoire) et COMA* (Cache Only Memory Access, où les mémoires locales se comportent comme des caches, i.e. une donnée n'a pas d'emplacement déterminé) ;
- mémoire distribuée* : dans ce deuxième cas, chaque processeur dispose de sa propre mémoire locale et d'une unité de calcul autonome. Les échanges d'information ne peuvent se faire que par échanges de messages sur le réseau d'interconnexion. On parle alors d'architectures de type DADM* (Disjoint Address space Distributed Memory), à accès mémoire de type NORMA* (NO Remote Memory Access).

7.1.3 Réseau d'interconnexion de machines à mémoires distribuées

Une machine multi-ordinateurs à mémoire distribuée est une architecture composée de plusieurs ordinateurs (processeur, mémoire, périphériques

2. La CM5, les CRAY T3D et CRAY T3E, l'ORIGIN 2000, la famille des Fujitsu VPP, le Sun HPC (Starfire) 10000 sont des exemples de machines MIMD...

d'entrées-sorties...) appelés noeuds, qui sont reliés par un réseau d'interconnexion permettant les échanges d'information par passage de messages. Les noeuds peuvent être soit complètement connectés (auquel cas chaque processeur est directement connecté avec tous les autres, cas du bus ou du crossbar) soit incomplètement connectés (ce qui implique l'existence d'une couche de routage pour assurer les connexions manquantes). D'autre part, les connexions dans le réseau peuvent être soit dynamiques (établies sur demande) soit statiques (permanentes). Dans le cas de connexions dynamiques, l'acheminement des messages est effectué par les points de commutation du réseau et dans les cas de connexions statiques, il est effectué par les processeurs eux-mêmes à l'aide de mécanismes de routage.

Il y a deux modes de routage correspondant chacun à un type de connexions :

- commutation de circuit : elle a lieu lorsque la liaison entre les unités communicantes est statique sur un circuit qui est établi avant la transmission du message ; les noeuds de commutation établissent le circuit lors du transfert ;
- commutation de paquets : elle a lieu lorsque la liaison entre les unités communicantes est dynamique. Le message, auquel des informations de routage ont été adjointes, établit son chemin pas à pas au sein du réseau.

Le second mode a un fonctionnement asynchrone (les opérations d'envoi sont non bloquantes, contrairement généralement au premier mode) et nécessite l'existence de tampons de stockage. Plusieurs messages transitent simultanément sur le même réseau. C'est le mode qui est le plus souvent utilisé dans les machines multi-ordinateurs à mémoire distribuée.

On représente traditionnellement un réseau d'interconnexion par un graphe simple non orienté dans lequel les sommets sont les processeurs et les arêtes les canaux de communication [Gengler et al., 1996, Chopard, 1998]. Puisque tout processeur peut communiquer avec n'importe quel autre processeur, le graphe du réseau d'interconnexion d'une machine à mémoire distribuée est toujours connexe. La dimension est la caractéristique principale du graphe ; elle permet de connaître le nombre de processeurs du réseau. Le diamètre est la longueur du plus grand des plus courts chemins reliant tout couple de processeurs (plus celui-ci est grand et plus ceci peut s'avérer pénalisant lors de phases de communications intensives entre deux processeurs éloignés). L'ensemble des sommets à distance 1 d'un sommet donné est appelé son voisinage. Le degré est le nombre de liens par processeur ; il permet de fixer la connectivité et de donner ainsi la complexité de communication en chaque noeud. Si ce nombre est constant, alors le graphe est régulier (la plupart du temps, on choisit un graphe régulier comme topologie du réseau de communication des machines). La largeur de bisection* d'un réseau de communication est le nombre d'arêtes qu'il faut "couper" lorsqu'on le scinde en deux le long d'un axe de symétrie pour produire deux sous-réseaux de communication isomorphes. Cette grandeur indique le nombre de liens disponibles pour relier une partie de la machine à l'autre. Plus elle est élevée et plus les possibilités de communiquer entre ces deux parties, sans provoquer de goulot d'étranglement, sont grandes. On appelle bande passante* (ou débit) la quantité d'information acheminée par unité de temps sur une arête du réseau de communication. Enfin, une mesure importante appelée latence* permet de donner pour une topologie donnée le temps nécessaire pour établir une connexion (temps de transfert d'une unité d'information dans le réseau).

Pour acheminer un message d'un processeur à un autre processeur distant dans les réseaux incomplètement connectés, on procède par routage matériel ou logiciel du message. Ce routage au sein du réseau est caractérisé par un mode de commutation du routeur. Ce mode de commutation qui correspond en fait à une gestion des trames au niveau des noeuds se fait selon deux modes :

- store and forward* : ce mode correspond à de la commutation de message. Les paquets sont stockés au niveau des noeuds intermédiaires dans leur globalité avant réémission vers le processeur suivant. Si ce routage évite les conflits d'accès aux liens, il ne peut cependant pas supprimer les risques d'interblocage pour l'accès aux tampons de stockage. De plus, il est relativement lent puisque la latence est proportionnelle au nombre de noeuds à franchir ;
- wormhole* : cette technique consiste à transférer les paquets au fur et à mesure de leur réception au niveau de chaque processeur. Ainsi, à une date donnée, un même message "occupe" plusieurs noeuds de communication (on parle de déplacement d'un ver, "worm", sur le réseau). Le principal problème lié à l'emploi de ce mode correspond à la gestion des conflits concernant les accès aux liens par plusieurs "vers" concurrents.

Si on adopte la stratégie "store and forward", alors le principe d'acheminement est le suivant : chaque noeud stocke chaque message entrant dans une mémoire tampon allouée dynamiquement. Si le destinataire du message est local, le message lui est délivré, sinon le message est envoyé sur un lien sortant. Il faut prévoir une stratégie de routage selon qu'il existe ou non plusieurs points de sortie vers le processeur destinataire. Cette stratégie peut être multiple : table de routage globale distribuée à chaque processeur, fonction de calcul permettant de connaître le lien cible au niveau local, tirage aléatoire du lien de sortie, diffusion inondante à chaque lien de sortie...

Après avoir présenté le parallélisme sous l'angle de l'architecture matérielle, abordons à présent le parallélisme d'un point de vue logiciel.

7.2 Généralités sur l'algorithmique parallèle

"Le parallélisme est une **technique** qui consiste à rechercher dans un problème les **tâches élémentaires** qui peuvent s'effectuer de manière **concurrente** afin d'optimiser le **temps total** d'exécution". Cette définition selon [ALMASI and GOTTLIEB, 1994] permet de présenter les quatre notions majeures qui lui sont rattachées et que nous allons aborder à présent. En l'occurence : les méthodologies de développement d'algorithmes parallèles, la notion de dépendance d'actions, les niveaux de parallélisme (ou plus exactement, les niveaux auxquels on recherchera des gains de performance par parallélisation) et la mesure du parallélisme.

7.2.1 Méthodologie du parallélisme

En ce qui concerne les techniques qui s'offrent à un développeur lorsqu'il souhaite paralléliser un problème, il peut, selon son environnement de programmation, procéder soit par parallélisation automatique (dans ce cas, le compilateur se charge de générer un code exécutable parallèle de façon automatique à partir d'un programme source écrit avec un langage classique comme le Fortran ou le C ; on se reportera ici plus particulièrement à [FEAUTRIER, 1995]), soit par parallélisation implicite (en utilisant un nouveau langage ou des extensions de langages classiques dédiés à l'expression de certaines formes de parallélisme comme HPF, CRAFT, OpenMP...), soit enfin par parallélisation explicite (en utilisant des additifs aux langages classiques qui permettent d'expliciter l'échange de messages ou la copie de mémoires à mémoires).

7.2.2 Dépendance d'actions

Pour déterminer le niveau de parallélisme d'un programme séquentiel, il importe d'arriver à produire, au sein de l'ensemble des instructions qui composent le code séquentiel de référence (que l'on cherche à paralléliser) les séquences d'action ou segments de programmes qui peuvent être réalisés de façon concurrente ou encore qui sont exécutables en parallèle. Pour ce faire, il faut déterminer si deux segments de code sont dépendants ou non[3] Les dépendances d'action ont été catégorisées comme suit :

- dépendance de données : il y a dépendance de données quand deux séquences d'action accèdent simultanément en écriture sur les mêmes variables (ou en lecture pour l'une et écriture pour l'autre) ou sur leurs références (pointeurs sur les variables),

- dépendance de contrôle : il y a dépendance de contrôle lorsque les séquences d'actions concurrentes sont ordonnancées au cours de l'exécu-

3. Ce dernier point est essentiel pour avoir une exécution déterministe.

tion du fait de branchements conditionnels, de boucles dont les indices sont réévalués à chaque pas...

- dépendance de ressources système : il y a dépendance de ressources système lorsque les actions réclament des ressources systèmes dont le nombre n'est pas suffisant pour les satisfaire toutes. Cette dépendance est souvent gérée par le système d'exploitation de la machine lui-même.

7.2.3 Granularité* de parallélisme

La granularité est une mesure de la quantité de calcul effectué par une action (qui sera soumise à parallélisation) d'un programme. Cette mesure dépend du niveau de programmation et est généralement exprimée en nombre d'instructions machine. La granularité désigne donc la taille moyenne des tâches élémentaires du point de vue du parallélisme. En pratique, il existe, pour simplifier, trois niveaux de granularité et l'on parle de parallélisme de grain fin, moyen ou gros.

- fine granularité : la recherche sur les séquences d'instructions susceptibles d'être indépendantes (et donc parallélisables) se fait au niveau de groupes d'une dizaine d'instructions au plus. L'avantage du calcul à grain fin est l'abondance de parallélisme propre au programme concerné. Les coûts de gestion associés à ce parallélisme font qu'il doit être pris en charge par le matériel lui-même pour être rentable ; les machines SIMD en sont le lieu bien souvent ;

- granularité moyenne : ce degré correspond au niveau appelé "tâche" qui regroupe les notions de procédures, routines, primitives... (de l'ordre du millier d'instructions). Ce parallélisme est du ressort du programmeur car les dépendances évoquées précédemment entre ces séquences d'intructions sont plus difficiles à évaluer par un compilateur. L'avantage de ce grain est son compromis entre l'efficacité de la parallélisation (qui reste correcte malgré son éloignement des ressources physiques) et l'adéquation avec le problème à traiter par le découpage en modules de taille raisonnable dont le parallélisme puisse être exploité sur les machines MIMD actuelles selon un modèle de programmation SPMD (Single Program Multiple Data) ;

- grosse granularité : ce degré correspond au niveau de l'application elle-même. Le nombre d'instructions n'est ici plus limité et le parallélisme est recherché dans le déroulement même des travaux ou dans la mise en concurrence des différentes applications ou des différents utilisateurs. Exception faite des ressources, il n'y a souvent pas de dépendance entre les programmes.

En ce qui concerne les questions de granularité on peut se reporter à [ARNOLD et al., 1992].

7.2.4 Les trois formes réalisables du parallélisme

En théorie, il est d'usage d'envisager un algorithme parallèle de trois façons différentes : réplication simultanée de tâches identiques sur des données de même nature, distribution d'actions concurrentes bien distinctes mais simultanées, ou enfin, enchaînement des tâches. Nous verrons avec nos simulations que, dans la pratique, les frontières entre ces trois formes de parallélisme sont à nuancer et que l'on peut combiner ces modes.

Parallélisme de données

Encore appelé data-parallélisme*, ce mode de programmation consiste à appliquer simultanément une même opération à des données de même type appartenant à une structure de données régulière. Dans ce type de parallélisme, les données sont associées aux ressources de calcul (réparties par décomposition de domaine sur chaque processeur) et l'exécution des tâches est uniforme. Cela correspond parfaitement au modèle d'architecture de machine SIMD à flot de contrôle centralisé (mais on peut cependant aussi faire du parallélisme de données sur une architecture de type MIMD en veillant à synchroniser l'ensemble des processeurs régulièrement de façon adéquate) et à une granularité généralement fine ou moyenne. Ce parallélisme est utilisé pour certaines classes de problèmes bien définies telles que le traitement d'image, le calcul matriciel...

Parallélisme de contrôle

Encore appelé parallélisme de tâches, ce mode de programmation consiste à créer plusieurs tâches élémentaires et à les distribuer sur les différentes unités de calcul, en tenant compte des dépendances de séquences liées à l'algorithmique même (gestion des synchronisations) et des communications interprocesseurs. Cette approche correspond à un modèle d'exécution MIMD (ou SPMD*, pour "Single Program Multiple Data", lorsque les différents processus exécutent simultanément le même programme) et à une granularité généralement moyenne ou grosse. Les différents éléments de calcul exécutent des programmes écrits dans un langage adapté ou dans un langage standard couplé avec une bibliothèque d'échanges de messages.

Parallélisme de flux

Dans ce dernier mode de programmation parallèle, les données à traiter traversent les tâches élémentaires en cascade. Les tâches sont elles-mêmes associées à des ressources de calcul chaînées entre elles en mode pipeline. Cela correspond à des modèles d'exécution variés (SIMD ou MIMD) et à des granularités quelconques, puisque c'est l'application qui fixe la nature de chaque "étage" du pipeline, qui peut alors être de tout type et toute taille. Cette approche conduit à la vectorisation dans le cas d'une fine granularité.

7.3 Evaluation du parallélisme

Après avoir présenté la méthodologie du parallélisme, abordons à présent son évaluation. Nous pouvons d'entrée, constater que la mesure du temps d'exécution n'est pas un critère suffisant et qu'il faut envisager d'autres critères plus informatifs. Supposons d'abord que la taille d'un problème puisse être représentée par un entier naturel n. Notons ensuite p la taille d'une machine parallèle, c'est-à-dire le nombre de ses processeurs. Nous noterons alors $T_p(n)$ le temps mis par une machine parallèle de p processeurs à résoudre un problème de taille n.

L'accélération* (ou speedup* en anglais) d'un algorithme est le rapport du temps pris par le meilleur algorithme séquentiel pour obtenir le résultat sur le temps d'exécution de l'algorithme parallèle par p processeurs.

$$S_p(n) = \frac{T_s(n)}{T_p(n)}$$

Lorsque le temps du meilleur algorithme séquentiel est inconnu, on définit la parallélisabilité* d'un problème de taille n par :

$$\mathcal{S}_p(n) = \frac{T_1(n)}{T_p(n)}$$

Comme généralement $T_1(n) > T_s(n)$, le critère de parallélisabilité offre une mesure plus favorable que celle du critère de l'accélération. Il indique alors par exemple, l'influence des surcoûts liés à la gestion même du parallélisme. En théorie, l'accélération (ou la parallélisabilité) d'un algorithme donné est un rationnel compris entre 1 et p. Si celle-ci est inférieure à 1, alors la parallélisation est catastrophique, et si elle est supérieure à p (ce qui est très rare, on parle alors de gains sur-linéaires ou encore supra-linéaires qui sont souvent dûs à des défauts de cache) on peut parfois en conclure que l'algorithme séquentiel comparé n'est pas optimal.

Tout comme dans la littérature où la confusion est parfois entretenue, nous décidons d'assimiler, pour l'ensemble de cette thèse, les notions de parallélisabilité et d'accélération. A parti de maintenant, nous n'utiliserons plus que le terme d'accélération tout en sachant qu'il ne s'agit souvent que de parallélisabilité.

On définit l'efficacité* d'un algorithme parallèle comme étant le rapport de son accélération sur le nombre de processeurs correspondant :

$$E_p(n) = \frac{S_p(n)}{p}$$

C'est une valeur comprise entre 0 et 1 que l'on exprime souvent en terme de pourcentage. Elle indique le taux de réussite de la parallélisation indépendamment du nombre de processeurs utilisés (note : si l'on ne connaît pas la valeur de l'accélération, on lui substitue la valeur de parallélisabilité dans l'équation ci-dessus). Tracer le graphe donnant l'efficacité d'un algorithme parallèle en fonction du nombre de processeurs permet de quantifier approximativement le surcoût dû au parallèlisme pour un nombre de processeurs donné.

En fait, l'accélération n'est pas forcément la mesure "idéale" du parallélisme. En effet, si l'on considère deux algorithmes A et B distincts permettant de résoudre un même et unique problème, il se peut que l'accélération de A soit bien supérieure à celle de B mais que, par ailleurs, le temps d'exécution de A soit bien supérieur à celui de B pour un nombre de processeurs donné disponibles. Asymptotiquement, A serait donc préférable à B... pour autant que le nombre de processeurs à mettre en jeu soit disponible !

Loi d'Amdahl*

Un algorithme donné comporte une partie séquentielle (ie une portion de code dont le temps ne peut être divisé même par addition de processeurs supplémentaires) et une partie parallèle (ie une portion de code dont le temps total est à diviser par le nombre de processeurs en jeu). Partant de ce constat, la loi d'Amdahl stipule que l'accélération globale d'un algorithme est majorée par un facteur dépendant de la partie séquentielle du code quand on fait tendre le nombre de processeurs de la machine parallèle vers l'infini.

$$\text{si } T_p(n) = T_p^{seq}(n) + T_p^{par}(n) = T_s^{seq}(n) + \frac{T_s^{par}(n)}{p}$$

$$\text{avec } T_s^{seq}(n) = f(n).T_s(n) \text{ et } T_s^{par}(n) = (1 - f(n)).T_s(n)$$

$$\text{alors } S_p(n) = \frac{1}{f(n) - \frac{1-f(n)}{p}} \overset{p\to\infty}{\longrightarrow} \frac{1}{f(n)}$$

Cette formule signifie que l'accélération est donc limitée par la partie séquentielle d'un algorithme. Si une fraction de 10% du code est purement séquentielle, l'accélération maximale sera inférieure ou égale à 10, quel que soit le nombre de processeurs utilisés [Gengler et al., 1996]. Paralléliser un problème essentiellement séquentiel peut s'avérer fortement inutile.

7.4 Nos choix architecturaux et logiciels propres

7.4.1 Architectures cibles

Le CRAY T3E

Il a succédé au CRAY T3D, qui fut la première machine massivement parallèle de CRAY Research Inc., à la mi-1996. C'est une machine indépendante (ou "self hosted") qui se vend avec une configuration allant de 16 à 2048 processeurs de calcul [DESPREZ, 1996b]. Ces processeurs sont des Dec Alpha 21164 Risc 64 bits dotés d'une mémoire locale allant de 64 Mo à 2 Go, de deux caches de données (un primaire de 8 Ko à adressage direct, direct mapped et un secondaire de 96 Ko [GIROU, 1995]) et d'un cache d'instructions de 8 Ko. Leur puissance de crête est de 600 Mflop/s. Dans sa configuration basse, cette machine comporte donc 1 Go de mémoire et a une puissance de crête de 9,6 Gflop/s, alors que dans sa configuration haute elle passe à 4 To de mémoire et 1,2 Tflop/s de puissance de crête. Son système d'exploitation est un Unicos MK 2.0 à micro-noyaux sur chaque processeur, basé sur Chorus, qui contrôle les communications entre processeurs, gère les allocations de mémoire, les interruptions et offre une interface système unique. Le réseau d'interconnexion du CRAY T3E est un tore 3D bi-directionnel, adaptatif. Le lien d'interconnexion entre les noeuds[4] est à faible latence ($1\mu s$) et forte bande passante (480 Mo/s).

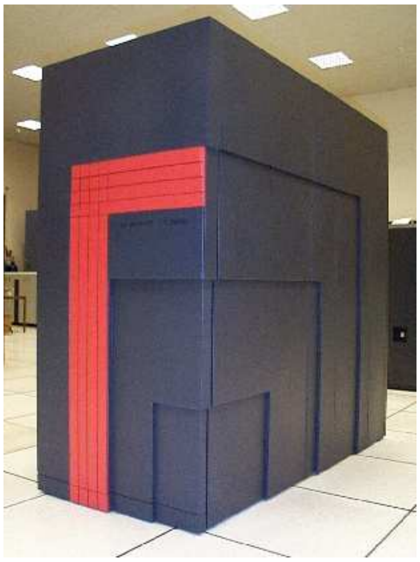

FIG. 7.1 – *Un* CRAY T3E

4. Un noeud est composé d'un microprocesseur, d'une mémoire locale et d'une interface réseau.

Toutes nos simulations ont été effectuées (pour ce qui est du traitement lui-même) sur le CRAY T3E de l'IDRIS [5]. C'est une machine dotée de 256 processeurs d'application, 7 processeurs système qui assurent la gestion globale du système et 9 processeurs de commande qui assurent la gestion de l'interactif et des tâches courantes de compilation et d'édition. Dans sa configuration actuelle, il n'autorise pas le temps partagé (ou "time sharing") et est accessible, à l'IDRIS du moins, en traitements interactifs ou en traitements par lots, avec le système NQS [6]. En interactif, un utilisateur est limité à 32 processeurs et 15 min de temps CPU par processeur. En traitement par lots, un utilisateur appartenant à la classe "développement" [7] est limité à 128 processeurs et 1h30 de temps CPU par processeur, au maximum et en nocturne [GIROU, 1995].

Le CRAY T3E de l'IDRIS est une excellente plate-forme d'exécution [8] dotée d'un bon outil de déboguage (Totalview) et d'un bon outil d'analyse des performances (Apprentice).

L'ORIGIN 2000

L'ORIGIN 2000 fait partie de la dernière génération de super-calculateurs à mémoire distribuée partagée (DSM: Distributed Shared Memory). Cette machine se vend avec une configuration allant de 1 à 128 processeurs 64 bits de calcul. Ces processeurs sont des R10000 (Mips 4) cadencés à 195 MHz, dotés de deux caches de données (un primaire de 32 Ko et un secondaire de 4 Mo) et d'un cache d'instructions de 32 Ko. Leur puissance de crête est de 390 Mflop/s [SILICON GRAPHICS, 1999a, SILICON GRAPHICS, 1999c, SILICON GRAPHICS, 1999b].

Ce serveur de calcul est basé sur une architecture S2MP (Scalable Shared-memory MultiProcessing). Cette architecture est constituée de composants modulaires reliés par des connexions de type CrayLink formant un hypercube. L'architecture de l'ORIGIN 2000 pour 16 noeuds et 32 processeurs est de la forme suivante :

Chacun des sommets du cube est un routeur (SGI Spider) relié à deux noeuds, qui est connecté à 3 routeurs voisins. Un noeud (ou carte nodale IP27) est composé de deux processeurs R10000, d'une mémoire partagée et d'un concentrateur (ou "hub"). Le concentrateur est connecté aux deux processeurs du noeud, il sert d'interface pour les entrées/sorties et les connexions avec le reste des processeurs de la machine. La mémoire est ainsi partagée par les deux processeurs du noeud à travers lui, mais également adressable

5. Institut du Développement et des Ressources en Informatique Scientifique, CNRS, Orsay.

6. Network Queuing System.

7. Il existe aussi une classe "production" offrant plus de possibilité...

8. D'après le 11e classement, datant du 18 juin 1998, des 500 machines les plus puissantes au monde il est situé au 34e rang mondial (cf http://www.top500.org/top500.list.html).

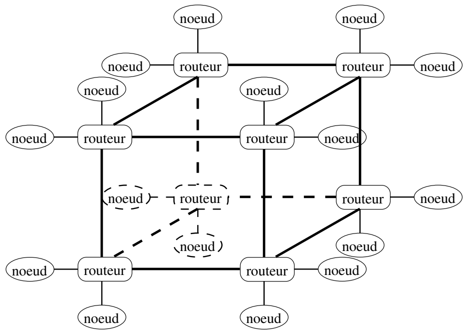

FIG. 7.2 – *Architecture simplifiée de l'*ORIGIN 2000 *à 16 noeuds et 32 processeurs*

par l'ensemble des processeurs de la machine. L'accès à la mémoire distante (au sens où elle est placée sur une autre carte nodale) se fait donc par le biais du CrayLink bidirectionnel à raison de 780 Mo/s [SILICON GRAPHICS, 1999a, SILICON GRAPHICS, 1999c, SILICON GRAPHICS, 1999b, LAUDON and LENOSKI, 1999], avec une latence moyenne de 773 ns dans le cas d'une configuration à 32 processeurs[9] [LAUDON and LENOSKI, 1999].

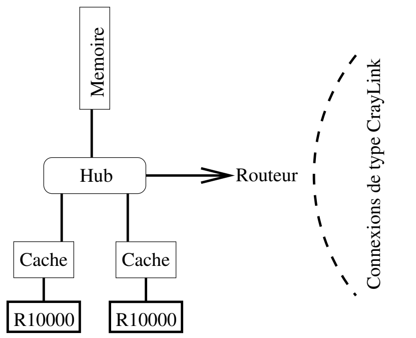

FIG. 7.3 – *Architecture simplifiée d'un noeud de l'*ORIGIN 2000 *à 32 processeurs*

Notre simulation bi-dimensionnelle a été portée (pour ce qui est du traitement lui-même) sur la machine du Pôle Parallélisme Ile-de-France Sud[10] qui, dans sa configuration actuelle, comprend 32 processeurs. La mémoire centrale est physiquement distribuée (512 Mo par noeud sous forme de 4 bar-

9. A titre de comparaison, on notera que la latence correspondant à l'accès au cache L1 est de 5,1 ns, que celle d'accès au cache L2 est de 56,4 ns et que celle d'accès à la mémoire locale est de 310 ns.

10. Ce super-calculateur est situé dans les locaux du Laboratoire de Mécanique et Technologie de l'ENS-Cachan.

rettes de 128 Mo de mémoire de type SDRAM*[11] montées sur des modules DIMM*[12]), mais elle est globalement visible et adressable par tous les processeurs (8 Go). Les 16 disques de plus de 9 Go chacun, offrent globalement un espace de stockage de 145 Go. Son système d'exploitation est un IRIX 6.4[13].

FIG. 7.4 – *L'ORIGIN 2000 à 32 processeurs du Pôle Parallélisme Ile-de-France Sud*

Le réseau de stations de travail

Il y a de nombreux avantages à réutiliser l'existant en matière de stations de travail raccordées en réseau : faibles coûts d'exploitation et de mise en place, environnement de travail habituel du développeur (le travail d'adaptation est donc limité), réseau extensible "à volonté"... Envisager de transformer un parc de stations de travail en une machine parallèle n'est donc pas absurde[14].

11. Synchronous Dynamic Random Access Memory
12. Dual Inline Memory Modules
13. Le passage en IRIX 6.5.3 a été fait le 4 mai 1999...
14. Avalon du Los Alamos National Laboratory est d'ailleurs une machine parallèle constituée de 68 processeurs DEC Alpha (ayant la distribution RedHat 5.0 de Linux pour système d'exploitation et la bibliothèque MPICH pour l'échange de messages...) et d'un commutateur Ethernet 3Com. Sa puissance, toujours d'après le site http://www.top500.org/top500.list.html, le place à pied d'égalité avec une ORIGIN 2000 à 64 processeurs de SGI dont le prix approximatif est de 1,8 million de dollars. D'après le 11e classement des 500 machines les plus puissantes au monde, datant du 18 juin 1998, il est situé au 315e rang mondial. Il n'a coûté que 150 mille dollars !

La difficulté de ce type d'approche réside dans la prise en compte de l'hétérogénéité. En effet, il est alors possible d'être confronté à une hétérogénéité des processeurs (jeu d'instructions, format et orientation des données en little Endian* ou big Endian* [15], format des adresses...), des architectures (du micro ordinateur séquentiel à la machine massivement parallèle...), des réseaux (avec des topologies, des protocoles, des performances et des composants différents), du système d'exploitation (PVM 3.4 permet à des plates-formes windows 32 bits comme Windows 95 et Windows NT de coopérer avec des plates-formes Unix [16] ; il existe par ailleurs une implémentation incomplète de MPI pour réseau de Macintosh G3 et MPICH existe sous Windows 3.1 et Windows NT [17]), des modèles d'exécution et des modèles de programmation, comme le recense [Desprez, 1996b].

Disposant d'un accès à une véritable machine parallèle, nous n'avons pour notre part pas cherché à utiliser le réseau des stations de travail de notre laboratoire comme plate-forme pour l'évaluation des performances (d'ailleurs déplorables étant donné qu'il s'agissait de temps partagé et d'un réseau Ethernet à 10 Mbit/s...). Par contre, il est vrai que celles-ci ont constitué une bonne plate-forme de test et de mise au point [18].

7.4.2 Bibliothèques d'échanges de messages utilisées

Dans le cas du modèle de programmation par échanges de messages, le programme est écrit dans un langage classique (tel que le langage C, Fortran, C++...) ; chaque processus exécute éventuellement des parties différentes du programme ; toutes les variables du programme sont privées et résident dans la mémoire locale de chaque processus et une donnée est échangée entre deux ou plusieurs processus par appel explicite à une primitive de la bibliothèque de communication au sein même du programme. C'est a priori la méthode de parallélisation la plus performante, relativement portable.

Comme le souligne [Raffin, 1997], la programmation par passage de message ne garantit pas le déterminisme des programmes. Ainsi, à partir de données initiales identiques, deux exécutions d'un même code peuvent ne pas donner les mêmes résultats. Cet indéterminisme peut-être conséquence de l'utilisation d'instructions de sélection de façon mal maîtrisée au sein du code, conjugué avec l'emploi d'instructions d'émission ou réception de mes-

15. D'après [Stern, 1995], une machine little Endian place les octets de poids faible d'un mot en premier, alors qu'une machine big Endian place ceux de poids forts en premier. Il importe de prendre en compte le fait qu'un PC sous Linux est little endian alors qu'une station Sun, elle, comme la majorité des plates-formes sous Unix est big endian.

16. d'après le site http://www.epm.ornl.gov/pvm.

17. d'après le site http://www.mcs.anl.gov/mpi.

18. Nous avons même utilisé un micro-ordinateur individuel pour la mise au point, puisque du point de vue logique et physique, une machine physique mono-processeur (séquentielle donc) peut simuler tout à tour l'exécution de plusieurs processus logiquement parallèles.

sages multiples (sans synchronisation ou sans identification de l'émetteur, par exemple). Par ailleurs, toujours d'après [RAFFIN, 1997], les programmes avec parallélisme par passage de message sont difficiles à appréhender car l'ordre syntaxique des instructions, dont la lecture séquentielle usuelle donne une vision intuitive, ne correspond souvent pas à l'ordre exact d'exécution. On peut par exemple trouver une instruction de réception de message avant l'envoi correspondant (du point de vue syntaxique), alors qu'à l'exécution la réception ne se termine pas tant que l'envoi n'a pas eu lieu.

Nous avons utilisé pour notre part les bibliothèques d'échanges de messages PVM et MPI. Il importe cependant de savoir qu'elles ne sont pas les seules à avoir été développées, et qu'il existe aussi d'après [DESPREZ, 1996c] : APPL (développée à NASA Lewis), CMMD (la bibliothèque native des machines de Thinking Machine Corporation), Chameleon (développée au Argonne National Laboratory), MPL (EUI-H, la bibliothèque native du SP-2 d'IBM), NX (la bibliothèque native des machines Intel), Parmacs et P4 (développées au Argonne National Laboratory), PICL (développée au Oak Ridge National Laboratory), Zipcode (développée à Livermore)...

La bibliothèque PVM [GEIST et al., 1994]

A l'été 1989, démarre un projet de recherche universitaire au Oak Ridge National Laboratory et en 1990 sort un prototype de version 1.0 qui n'a jamais été rendu public. En février 1992 est publiée une version 2.0 développée par l'Université du Tennessee. Tout au long de l'année 1992 sortent les versions 2.1 à 2.4 qui apportent des modifications et des corrections à la version 2.0. En 1993, sort la version 3.0 qui sera réécrite complètement, modifiée et étendue jusqu'à la version 3.3.11 (qui est celle que nous avons utilisée). En 1997, la version 3.4 marque un tournant de cette bibliothèque d'échanges de messages avec l'introduction d'extensions majeures comme les contextes de communication et les groupes statiques de processus.

La machine parallèle PVM est une architecture virtuelle qui permet de réaliser un parallélisme asynchrone (non bloquant), distribué, avec réseau de communication. D'après [COUSOT, b], il est possible de relier par son biais jusqu'à 4095 machines exécutant au plus $2^{18}-1$ tâches[19] chacune. Ces machines peuvent être hétérogènes et communiquer par messages au travers d'un réseau hétérogène. Pour être opérante, la machine parallèle PVM doit lancer au préalable un processus démon `pvmd3` (cas d'un système Unix) ou un processus démon `rshd` ou `rexecd` (cas d'un système Windows 32 bits) sur chaque machine physique et pour chaque utilisateur. Le ou les programmes applicatifs doivent être positionnés sur chaque système de fichiers (on peut

19. Le terme processus est réservé au monde Unix.

en effet faire du montage NFS* [20]) dans le bon répertoire, et l'utilisateur doit être authentifié et autorisé à se connecter sur chacune des plates-formes (on peut aussi faire un montage NIS* [21]). Une console PVM permet d'assurer un suivi de l'évolution des applications sur la machine parallèle virtuelle, mais elle permet aussi de modifier la configuration de la machine par ajout ou suppression de noeuds de calcul.

Une application PVM, c'est un ensemble de tâches (ou processus) autonomes exécutant chacunes leur propre code et communiquant via des appels aux routines de PVM. Il existe des routines [22] de gestion de l'environnement, de communication point à point, de communication collective et de gestion des groupes. Ces routines sont regroupées dans trois bibliothèques de primitives : libpvm3.a, libfpvm3.a et libgpvm3.a qui correspondent respectivement aux routines C, aux routines Fortran et aux routines de gestion des groupes dynamiques.

Gestion des entrées-sorties : Il n'y a pas d'entrée standard en PVM, il faut donc se contenter de lire des fichiers accessibles en lecture. Les sorties sont, par défaut, redirigées vers la console pour le processus père ou vers des fichiers de trace des machines PVM respectives pour les processus fils [Mignot et al., 1996, Cousot, a]. Les sorties des processus fils peuvent cependant être rappatriées sur la console du processus père à l'aide de la primitive `int pvm_catchout()`.

Contrôle des tâches : une tâche PVM possède un identifiant global unique, appelé `tid`, au sein de la machine virtuelle. Cette tâche est exécutée sous forme d'un processus qui possède, lui, un identifiant local unique, appelé `pid`, propre à la machine physique sur lequel il s'exécute. Le `tid` propre à chaque tâche permet une désignation sans ambiguïté lors de l'échange de messages. Nous donnons ci-après les noms des principales primitives de la bibliothèque en langage C. En Fortran, les noms sont très voisins.

`int pvm_mytid()`	connection avec le démon local pvmd3 (enrôlement de la tâche)
`int pvm_exit()`	déconnection avec le démon local pvmd3 (la tâche redevient simple processus Unix)

20. Le Network FileSystem est un système qui fournit un accès transparent aux fichiers dispersés sur les disques d'un parc de machines hétérogènes où fonctionnent des systèmes d'exploitation différents.

21. Le Network Information System est un système qui fournit un service de base de données réparti simplifiant la gestion des fichiers administratifs les plus importants d'un parc de machine, comme le ficher des mots de passe. Quand NIS est installé la modification de ces fichiers sur le serveur maître permet ensuite, par simple propagation automatique, d'éviter d'avoir à modifier les fichiers de tous les systèmes du réseau.

22. Plus d'une soixantaine...

`int pvm_spawn()`	lancement de tâches filles à partir de la tâche courante
`int pvm_parent()`	récupération du tid de la "tâche mère" (ou PvmNoParent sinon)

Communications entre tâches, envoi de messages : un message c'est une enveloppe constituée de l'identificateur de tâche de l'émetteur, de l'identificateur de tâche du récepteur, du `tag` ou étiquette du message, du type et de la taille des données à transférer et d'un contenu : le message lui-même. Les communications sous PVM sont asynchrones et peuvent être bloquantes ou non. Elles préservent l'ordre des messages[23] et peuvent avoir pour destinataires l'ensemble des tâches (multicast*) ou un groupe de tâches (broadcast*, il faut au préalable avoir défini un groupe dynamique).

`int pvm_initsend()`	initialisation du tampon d'émission et spécification du type d'encodage des données (encodage XDR*, eXternal Data Representation, ou de non encodage, possibilité d'éviter la recopie du buffer d'émission)
`int pvm_pk[type]()`	rangement des données (éventuellement hétérogènes) dans le tampon d'émission
`int pvm_send()`	émission asynchrone d'un message
`int pvm_recv()`	réception bloquante d'un message
`int pvm_nrecv()`	réception non bloquante d'un message
`int pvm_upk[type]()`	extraction des données reçues, une par une, du tampon de réception
`int pvm_mcast()`	émission multicast d'un message
`int pvm_bcast()`	émission broadcast d'un message
`int pvm_scatter()`	distribution sélective de données au sein d'un groupe dynamique
`int pvm_gather()`	collecte sélective de données au sein d'un groupe dynamique
`int pvm_reduce()`	réduction répartie sur un groupe dynamique

Gestion des groupes dynamiques et synchronisation des tâches : Les groupes de processus permettent de partitionner l'ensemble des processus. A l'enrôlement d'une tâche dans un groupe dynamique, celle-ci se voit attribuer un identifiant appelé gid qui correspond à son "rang" dans le groupe. Ce rang est un entier positif qui est incrémenté d'une unité à chaque nouvel enrôlement.

23. Dans le sens où deux messages envoyés successivement d'une même source vers une même cible, arrivent à destination dans l'ordre où ils ont été émis.

`int pvm_joingroup()`	incorporation d'une tâche dans un groupe dynamique
`int pvm_lvgroup()`	exclusion d'une tâche d'un groupe dynamique
`int pvm_gsize()`	nombre exact de tâches inclues dans un groupe dynamique
`int pvm_gettid()`	obtention du tid d'une tâche de gid donné, au sein d'un groupe
`int pvm_getinst()`	obtention du gid d'une tâche de tid donné, dans un groupe donné
`int pvm_barrier()`	synchronisation des tâches d'un groupe donné

Spécificités de PVM sur le CRAY T3E : Sur le CRAY T3E, l'emploi de la bibliothèque PVM est un peu particulier [CHERGUI and DUPAYS, 1995]. En effet, le modèle de programmation est SPMD (tous les processeurs exécutent un même code qui peut cependant comporter des branchements conditionnels, il est donc toujours possible d'émuler le modèle MPMD*[24]) et il n'est pas possible de créer des tâches filles par programme (faire du **`pvm_spawn()`** programmé). Par ailleurs, il n'y a pas à lancer de démon pvmd3, et la console pvm n'est pas accessible à un utilisateur quelconque donné. Enfin, du point de vue de la gestion des groupes, pour chaque application PVM, il existe, pendant toute la durée de son exécution, un groupe statique dénommé PVMALL[25] qui englobe toutes les tâches.

La bibliothèque MPI [SNIR et al., 1995]

L'histoire de cette bibliothèque démarre en avril 1992, à Williamsburg en Virginie, dans un groupe de travail intitulé "Standards for Message Passing in a Distributed Memory Environment". C'est là qu'est initié le processus de création de ce standard et qu'est créé un premier groupe de travail. En novembre 1992, Dongarra, Hempel, Hey et Walker sortent une version préliminaire de MPI. Celle-ci est retravaillée par les membres du "Message Passing Interface Forum" jusqu'en avril 1994 et le 5 mai 1994 est publiée la version 1.0[26] des spécifications de cette bibliothèque. Des modifications et des corrections mineures ont ensuite successivement donné naissance aux versions 1.1[27] et 1.2 du document. En avril 1995, le "MPI Forum" commence à envisager d'étendre les possibilités de cette bibliothèque et le 25 avril 1997 sont publiées les spécifications de la version 2.0. Cette nouvelle version enrichit la précédente de fonctionnalités telles que la gestion dynamique des

24. Pour "Multiple Program, Multiple Data".
25. Et qui correspond à la chaîne de caractères vide.
26. Une version PostScript du document est disponible à l'adresse http://www.mpi-forum.org/docs/mpi-10.ps.
27. Une version PostScript du document sorti le 12 juin 1995 est disponible à l'adresse http://www.mpi-forum.org/docs/mpi-11.ps.

processus (fonctionnalité déjà présente dans PVM mais pas dans MPI 1.2 où il n'y a pas de `MPI_Spawn`), les communications mémoire à mémoire (il existe déjà sur CRAY T3E une bibliothèque propriétaire de copie de mémoire à mémoire : ShMem), les entrées-sorties parallèles, les extensions "temps-réel", les communications collectives non bloquantes, les liaisons avec les langages Fortran 90 et C++ [CHERGUI et al., 1997, MIGNOT et al., 1996].

Une application MPI, c'est un ensemble de processus autonomes exécutant chacun leur propre code et communiquant via des appels aux routines de MPI. Il existe des routines [28] de contrôle des processus et gestion de l'environnement, de communication point à point, de communication collective, d'écriture de types de données dérivés, de gestion des communicateurs et des topologies virtuelles de processus. L'unité de calcul fondamentale est le processus [DESPREZ, 1996c] et chaque processus possède un flot de contrôle indépendant et un espace d'adressage séparé. Il n'y a pas en MPI-1 de mécanisme pour assigner un processus à un processeur donné, ou pour créer ou détruire des processus. Cependant MPI supporte les groupes de processus dynamiques qui peuvent être créés ou détruits, se recouvrir...

Contrôle des processus et gestion de l'environnement :

`MPI_Init`	initialisation de l'environnement MPI
`MPI_Finalize`	désactivation de l'environnement MPI
`MPI_Wtime`	différence entre la date actuelle et une date de référence

Communications entre processus, envoi de messages : Dans MPI, un processus est spécifié par un nom de groupe et un rang (ou numéro d'instance, entier naturel) relatif dans le groupe correspondant. A chaque message est associé une étiquette qui est spécifiée par un contexte de message et un indice de message relatif au contexte. En fait, les groupes partitionnent l'espace des processus, tandis que les contextes partitionnent l'espace des étiquettes de messages. On associe les groupes et les contextes pour former des objets appelés communicateurs (cette dernière notion masque celle des contextes qui ne sont pas visibles au niveau de l'applicatif MPI). Un point important de MPI, réside dans le fait de pouvoir communiquer avec un processus fictif de rang `MPI_PROC_NULL`. Par ailleurs, à la réception d'un message, le rang du processus source et l'étiquette du message peuvent rester indéfinis : `MPI_ANY_SOURCE` et `MPI_ANY_TAG`.

28. Exactement 128 dans la version 1.0 de MPI d'après [SNIR et al., 1995] !

`MPI_Send`	émission bloquante d'un message avec recopie du message dans une mémoire locale tampon avant envoi (mode standard, l'envoi peut-être démarré même si la réception correspondante n'a pas été initiée)
`MPI_[R,S,B]send`	émission bloquante d'un message en mode "ready" (l'envoi du message ne peut commencer que si la réception a été initiée auparavant), "synchroneous" (l'envoi ne se termine que si la réception a été postée et la lecture du message terminée) ou "buffered" (la recopie dans un buffer local est à la charge du programmeur, l'envoi et la réception sont totalement découplés)
`MPI_Recv`	réception bloquante d'un message
`MPI_I[r,s,b]send`	émission non bloquante d'un message en modes standard, ready, synchroneous ou buffered
`MPI_Irecv`	réception non bloquante d'un message
`MPI_Probe`	test bloquant de contrôle d'un message avant réception (pour identification de son étiquette ou de son émetteur)
`MPI_Bcast`	émission broadcast d'un message au sein d'un communicateur
`MPI_Gather`	collecte sélective de données réparties au sein d'un communicateur
`MPI_Scatter`	diffusion sélective de données réparties au sein d'un communicateur
`MPI_Reduce`	opération de réduction répartie au sein d'un communicateur
`MPI_Allreduce`	opération de réduction répartie suivie d'une diffusion broadcast du résultat au sein d'un communicateur

Types de données dérivés : Un message contient un certain nombre d'éléments de types particuliers. Ces types peuvent être soit pré-définis ou primitifs (appelés types de base de MPI[29], comme : `MPI_CHAR`, `MPI_SHORT`, `MPI_INT`, `MPI_FLOAT`, `MPI_DOUBLE`...), soit dérivés (ils sont alors construits à partir de types de base ou d'autres types dérivés à l'aide des primitives que nous présentons ci-après).

`MPI_Type_contiguous`	construction d'un type dérivé à partir de données homogènes, contiguës en mémoire

29. Attention les types de base du C sont différents de ceux du Fortran !

`MPI_Type_vector`	construction d'un type dérivé à partir de données homogènes, contiguës par blocs de données distants de pas constants
`MPI_Type_indexed`	construction d'un type dérivé à partir de données homogènes, contiguës par blocs de données distants de pas variables
`MPI_Type_struct`	construction d'un type dérivé à partir de données hétérogènes, contiguës par blocs de données distants de pas variables
`MPI_Type_commit`	validation d'un type dérivé après construction
`MPI_Type_free`	libération d'un type dérivé après utilisation
`MPI_Type_extent`	obtention de la taille étendue d'un type MPI aligné en mémoire
`MPI_Type_size`	obtention de la taille d'un type de données MPI non aligné en mémoire
`MPI_Address`	obtention de l'adresse d'une variable en mémoire

Les communicateurs : lorsqu'on lance une application MPI, il existe un communicateur par défaut appelé `MPI_COMM_WORLD`, qui comprend tous les processus actifs de l'application. Un communicateur (notion qui regroupe celle des groupes et des contextes comme nous l'avons vu précédemment) sert à créer un "univers de communication" indépendant ; il définit l'étendue de l'opération de communication.

`MPI_Comm_create`	création d'un nouveau communicateur à partir d'un groupe de processus
`MPI_Comm_size`	obtention du nombre de processus gérés par le communicateur donné en argument
`MPI_Comm_rank`	obtention du rang du processus dans le communicateur donné en argument
`MPI_Comm_dup`	duplication d'un communicateur
`MPI_Comm_split`	partitionnement d'un communicateur
`MPI_Comm_free`	libération d'un communicateur
`MPI_Barrier`	synchronisation globale de tous les processus du communicateur

Topologies virtuelles de processus : dans la plupart des applications du parallélisme et plus particulièrement, dans toutes les méthodes de décomposition de domaine, il est intéressant de pouvoir disposer les processus suivant une topologie régulière. MPI permet de définir des topologies virtuelles de processus de type "cartésien" (grille régulière périodique ou non de processus), ou de type "graphe" (topologie plus complexe) [Gropp, 1995, Pacheco, 1995, Brugeas, 1996].

`MPI_Cart_create`	création d'une topologie cartésienne sur le communicateur courant
`MPI_Cart_sub`	partitionnement (ou dégénérescence) d'une topologie cartésienne
`MPI_Dims_create`	obtention du nombre de processus dans chaque dimension dans le cadre d'une topologie cartésienne
`MPI_Cart_rank`	obtention du rang du processus en fonction de coordonnées dans la topologie cartésienne
`MPI_Cart_coords`	obtention des coordonnées d'un processus dans la topologie cartésienne associée en fonction de la donnée de son rang
`MPI_Cart_shift`	obtention du rang des processus voisins selon un axe donné de la topologie cartésienne
`MPI_Cart_get`	obtention d'information sur la topologie courante et le processus courant
`MPI_Graph_create`	création d'une topologie de type "graphe" sur le communicateur courant

7.4.3 Stratégies d'optimisation possibles

L'optimisation des communications dans le cadre de la communication par échanges de messages peut s'accomplir à différents niveaux :

- recouvrir les temps de communication par des temps de calcul en tenant compte du fait que le réseau et les processeurs opèrent de manière asynchrone (préférer donc les envois et réceptions de message non bloquants),
- éviter les recopies de messages dans des espaces mémoire temporaires (buffering, préférer donc le mode d'envoi synchroneous aux autres modes),
- minimiser les surcoûts induits par des appels répétitifs aux primitives de communication (de l'intérêt des modes de communications persistantes[30]).

Enfin, comme le constate [Desprez, 1996c], dans les problèmes de décomposition de domaine, "les communications apparaissent aux frontières. [Il faut donc] minimiser le rapport surface/volume".

30. qui ne sont cependant pas implémentés sur le Cray t3e [Chergui et al., 1997] !

Chapitre 8

Simulation parallèle uni-dimensionnelle

Lorsque l'on s'attaque au problème de la mise en œuvre informatique d'un modèle théorique, deux attitudes peuvent être envisagées : réutilisation d'un logiciel existant avec adaptation aux contraintes du problème particulier ou re-développement complet d'une plate-forme logicielle[1]. Dans ce chapitre, nous constatons que le problème de la représentation visuelle des résultats nous a fortement influencé dans le choix même d'un développement spécifique plutôt qu'une utilisation de simulateurs déjà développés. Ce choix se trouve d'ailleurs renforcé par la stratégie d'optimisation parallèle que nous développons juste après. Non pas qu'une parallélisation en espace soit novatrice, mais bien parce que nous avons une approche du transfert de l'information visuelle relativement spécifique. Enfin, après avoir développé la stratégie de sous-structuration que nous avons employée (avec équi-répartition de la charge des sous-domaines sur l'ensemble des processeurs et minimisation de la taille des problèmes aux interfaces), nous abordons l'implémentation, la portabilité et les spécificités propres au programme que nous avons écrit.

8.1 Représentation visuelle des résultats

Comme le note [Schatten, 1997], un réseau d'automates cellulaires uni-dimensionnel a, en théorie, le grand avantage sur un réseau d'automates cellulaires multi-dimensionnel d'être facilement visualisable : "the states of one timestep are plotted in one dimension, and the dynamic development can be shown in the second dimension". En effet, il suffit, par exemple, de représen-

1. Dans certains cas, on peut aussi bien sur envisager une solution intermédiaire avec incorporations (en plus ou moins forte proportion) de composants logiciels préexistants, dans un code que l'on développe spécifiquement. . .

ter la dimension d'espace en abscisse et la dimension temps en ordonnée[2]. Ceci dit, ce type de représentation visuelle est relativement peu adapté aux automates cellulaires ayant un nombre d'états supérieur à quelques unités (ou ayant plus d'un "attribut de couleur") et ce même dans le cas d'un réseau d'automates cellulaires uni-dimensionnel.

Ainsi, nous sommes confrontés à ce deuxième cas de figure, puisqu'il s'agit, partant d'un réseau d'automates cellulaires en une dimension d'obtenir une série de "clichés" en deux dimensions. Ces clichés doivent correspondre à la zone de subduction aux divers stades successifs de la simulation. Partant donc des sept attributs d'altitude de chacune des cellules du réseau à chaque pas de temps d'échantillonage, nous allons donc devoir reconstituer les différents clichés.

Initialement, nous avions choisi un mode de visualisation conversationnel ou temps réel (en ce sens qu'il n'est pas différé mais produit à chaque pas de temps) en utilisant une librairie graphique sous X11 développée par R. Cazoulat [Cazoulat, 1996b, Cazoulat, 1996a] de l'Université de Caen. Bien que pratique pour la mise au point de la fonction de transition, ce mode s'est toutefois vite avéré très lent, peu portable et inadapté au traitement parallèle. Par ailleurs, nous avons jugé que la visualisation sous forme d'une animation tient plus du post-traitement graphique que du traitement par automates cellulaires à proprement parler.

Nous avons donc finalement opté pour un post-traitement graphique d'images au format "raw" (RGB) que nous transformons en un film MPEG en utilisant l'utilitaire convert* du logiciel ImageMagick* du domaine public. Ce post-traitement a l'avantage d'être plus rapide à la consultation et plus portable puisqu'il suffit de disposer d'un "Mpeg player" (logiciel du domaine public) pour visualiser les animations.

Au cours du traitement, toutes les altitudes du tableau de cellule courant (et uniquement les altitudes puisque, en ce qui concerne la représentation visuelle des résultats, seuls ces attributs nous intéressent) sont périodiquement stockées dans un tableau de résultat global. En fin de traitement, ce tableau contenant l'ensemble des altitudes pour l'ensemble des "clichés" sélectionnés est enregistré sous forme d'un unique fichier binaire non formaté conservé sur disque.

Ces données brutes subiront alors successivement deux post-traitements graphiques : une première conversion au format RGB (accompagnée d'une traduction des 7 attributs d'altitude sous forme d'un tableau bi-dimensionnel directement visualisable avec un client graphique adapté) et un second traitement visant à transformer les différents fichiers RGB et à créer une animation graphique au format MPEG.

2. C'est de cette façon, par exemple, qu'avec un réseau d'automates cellulaires uni-dimensionnel ayant une fonction de transition locale adéquate, l'on peut reproduire la structure fractale de la parité dans le triangle de Pascal. . .

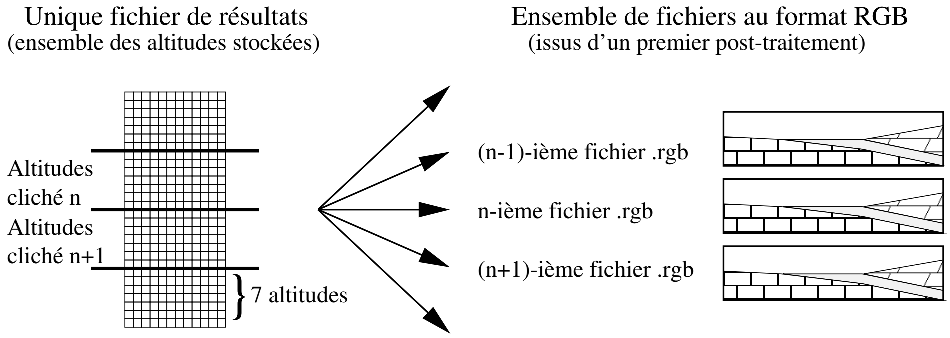

FIG. 8.1 – *Le premier post-traitement consiste à générer un ensemble de fichiers au format RGB à partir du tableau des altitudes des états des différentes cellules du réseau enregistrées périodiquement.*

8.2 Décomposition de domaine et stratégie d'optimisation parallèle

Un réseau d'automates cellulaires a, par définition, une nature intrinsèquement parallèle en espace. Ainsi, pour [VICHNIAC, 1984] : «since all the cells "compute" their new state simultaneously, cellular automata are often seen as a paradigm of distributed computation». Pour établir une stratégie d'optimisation parallèle de notre simulation, nous nous sommes inspirés des techniques de parallélisation en espace en vigueur dans d'autres domaines, comme la convolution en traitement d'image, les schémas explicites en mécanique... Ainsi, implémenter un automate cellulaire sur machine parallèle revient en quelque sorte à mettre en œuvre une résolution par discrétisation d'un problème d'équation aux dérivées partielles par un schéma explicite centré sur un maillage cartésien régulier. Ce schéma doit alors être appliqué sur un domaine de calcul rectangulaire avec un maillage de pas constant en espace comme en temps, et avec des conditions aux limites homogènes. Nous allons donc utiliser une méthode dite de décomposition de domaine*.

Comme le note [ACKLAM and LANGTANGEN, 1998], l'idée qui est à la base de la méthode par décomposition de domaine est qu'un problème global peut être considéré comme un ensemble de sous-problèmes locaux portant sur les partitions du domaine global. Une telle méthode favorise donc l'exploitation de la localité des données et offre la possibilité que tous les sous-problèmes soient résolus concurremment. Il importe bien sur que les traitements à appliquer aux divers sous-problèmes soient de même nature mathématique que ceux qui doivent être appliqués au problème global. En raison de la quasi-indépendance mutuelle des problèmes locaux, leurs résolutions (les itérations en temps) peuvent être affectées à un processeur de la machine parallèle utilisée. On peut de cette façon espérer un accroissement significatif des performances globales.

On appelle domaine de dépendance* d'une cellule du réseau l'ensemble des cellules de son voisinage ayant une influence[3] sur elle à une date quelconque donnée. Si l'on considère l'ensemble des automates cellulaires d'un sous-domaine quelconque, on constate que son domaine de dépendance s'étend à des cellules situées dans les deux sous-domaines voisins. Or, à chaque itération en temps, chaque processeur doit disposer de toutes les informations nécessaires à la résolution du problème sur la partie de domaine qui lui est affectée. On en déduit qu'il y a nécessité, pour chaque processeur, de communiquer à ses voisins immédiats les cellules situées aux interfaces.

D'un point de vue pratique, nous allons appliquer la méthode de décomposition de domaine avec recouvrement*. Le domaine de calcul global est donc partagé exhaustivement en plusieurs sous-domaines avec des zones de recouvrement. Dans le cadre d'une distribution uni-dimensionnelle des données, chaque sous-domaine doit envoyer ses cellules aux frontières aux processeurs voisins et, en retour, recevoir leurs propres cellules. Cette stratégie de sous-structuration implique qu'à chaque itération, chaque processus stocke des cellules n'appartenant pas à son propre sous-domaine (avant de les utiliser pour mettre à jour les cellules aux bords du sous-domaine). Pour résoudre ce problème de stockage local d'informations non-locales, nous avons introduit dans le modèle une zone de recouvrement de cellules fantômes* (encore appelées "ghost cells*" ou "guard cells*" dans la littérature). Ces cellules qui n'ont pas "d'existence réelle" vis-à-vis de la modélisation sont, d'un point de vue informatique, stockées sur le pourtour du sous-domaine de cellules du processus courant. Elles étendent ainsi d'une unité les cellules qui sont situées sur les "bords intérieurs" du sous-domaine courant. Cette stratégie implique aussi que les cellules fantômes soient échangées à chaque pas de temps afin d'être réactualisées simultanément avec l'ensemble des cellules situées à l'intérieur du sous-domaine considéré. A chaque itération en temps, chaque processeur dispose ainsi de toutes les informations nécessaires à la résolution du problème sur la partie de domaine qui lui est affectée.

L'important du point de vue de l'optimisation du temps de calcul est que les processeurs ne perdent pas trop de temps à communiquer à leurs voisins leurs cellules aux bords ou à attendre que leurs voisins leur aient communiqué les nouvelles valeurs des cellules fantômes. Nous avons donc fait en sorte qu'il y ait un recouvrement des temps de communication et des temps de calcul dans la portion de code correspondant à l'application de la fonction de transition du réseau d'automates cellulaires. Ainsi, chaque processeur effectue un envoi non-bloquant des cellules du "bord interne" et enchaîne immédiatement (sans se soucier de leurs réceptions par les proces-

3. Au sens où nous l'avons définie dans le chapitre d'introduction au formalisme concernant les automates cellulaires.

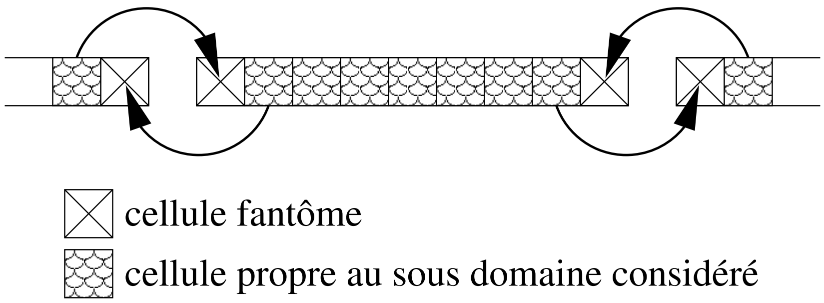

FIG. 8.2 – *Echange des cellules "fantômes" aux interfaces des sous-domaines.*

seurs voisins) sur la mise à jour des cellules de son sous-domaine propre qui ne nécessitent pas la connaissance d'état de cellules mises à jour au niveau des deux sous-domaines voisins. A la réception bloquante des cellules de l'interface en provenance des deux sous-domaines voisins, l'itérateur termine la mise à jour des cellules situées au bord de son sous-domaine propre.

En ce qui concerne le choix du moyen logiciel à proprement parler, nous avons constaté que le problème du maintien de l'adéquation des codes parallèles développés avec les matériels disponibles se pose de manière cruciale, puisque le parc des calculateurs parallèles se renouvelle fréquemment [4]. Nous avons donc cherché à nous affranchir de ces transformations matérielles en utilisant un produit de parallélisme pur, stable dans le temps et garantissant une relative autonomie vis à vis de la machine cible et donc une facilité au portage, comme l'est la bibliothèque d'échanges de messages PVM. Pour le type de modèle de communication par échanges de messages, nous avons opté pour un envoi de message asynchrone et une réception de message synchrone.

8.3 Algorithme simplifié

NBC désigne le nombre de cellules du tableau de cellules et *nbproc* le nombre de processus de l'application. *nbproc* doit-être un diviseur de NBC. Nous appelons "superviseur" le premier des processus à s'être inclus dans le groupe dynamique des "itérateurs".

4. J'ai assisté ainsi à la disparition de la CM5 du CNCPST et du CRAY T3D de l'IDRIS au cours de mes deux premières années de thèse.

L'algorithme parallèle

```
# Fonction de transition globale:
procédure $F_{globale}$(c,cc,$f_{locale}$,$Coeff$)
   cc ← c
   Envoyer les valeurs de $cc_1$ et $cc_n$ aux co-latéraux
   Pour $i \leftarrow 2$ à $n$, faire:
      $c_i.f_{locale}(cc_{i-1}, cc_i, cc_{i+1})$
   Réceptionner les valeurs de $cc_0$ et $cc_{n+1}$ en provenance des co-latéraux
   Calculer $c_1.f_{locale}(cc_0, cc_1, cc_2)$ et $c_n.f_{locale}(cc_{n-1}, cc_n, cc_{n+1})$
fin de procédure


# Programme principal:
Initialiser PVM
Prendre connaissance du nombre "nbproc" de processus clients
Chaque client s'inclut dans le groupe des itérateurs
si le processus courant est le superviseur alors
   lire le fichier des paramètres
   diffuser les valeurs de ces paramètres à l'ensemble des itérateurs
sinon
   réceptionner les valeurs des paramètres en provenance du superviseur
   fin si
soit c et cc deux tableaux de $n+2$ cellules (avec $n = \frac{NBC}{nbproc}$ et $n, NBC \in \mathbb{N}$).
Initialiser le tableau local c
Itérer le bloc d'instructions suivant:
   $F_{globale}$(c,cc,$f_{lente}$,$Coeff_{Tr}$)
   Collecter et additionner les valeurs de $Coeff_{Tr}$ de l'ensemble
   des processus itérateurs au niveau du superviseur
   si le processus courant est le superviseur alors
      diffuser la valeur de $Coeff_{Tr}$ à l'ensemble des itérateurs
   sinon
      réceptionner la valeur de $Coeff_{Tr}$ en provenance du superviseur
   fin si
   Itérer le bloc d'instructions suivant:
      $F_{globale}$(c,cc,$f_{moyenne}$,$Coeff_{Tr}$)
      Itérer le bloc d'instructions suivant:
         $F_{globale}$(c,cc,$f_{rapide}$,0)
      Tant que condition rapide non réalisée
   Tant que condition moyenne non réalisée
   si la condition de sauvegarde est réalisée alors
      collecter les tableaux locaux de résultats au niveau du superviseur
      si le processus courant est le superviseur alors
         sauver le cliché courant sur disque
      fin si
   fin si
Tant que condition lente non réalisée
Sortir de PVM.
```

Chapitre 9

Simulation parallèle bi-dimensionnelle

Dans ce chapitre, nous présentons l'implémentation de notre modèle bi-dimensionnel sur machine parallèle. Notre technique de simulation parallèle en espace (par décomposition de domaine) nous permet, par la même occasion, de justifier notre choix de répliquer les cellules aux bords et non de procéder par simple congruence modulo le nombre des cellules dans l'algorithme séquentiel, comme nous l'expliquions dans le chapitre concernant le modèle bi-dimensionnel. Nous verrons dans le chapitre suivant que le portage de notre code sur une plate-forme parallèle à mémoire distribuée nous a permis d'optimiser le temps d'exécution de notre simulation.

9.1 De l'intérêt de développer notre propre plate-forme logicielle

Il existe déjà, dans le domaine public, des outils permettant de visualiser [Creutz, 1995] ou mettre en œuvre [Eckart, 1997] des réseaux d'automates cellulaires. Ainsi, nous nous sommes servis du Cellular Automata Simulation System* [Eckart, 1997] pour la mise au point de la fonction de transition du modèle bi-dimensionnel, qui est très pratique du fait de son mode de visualisation conversationnel. Pourtant, tout comme pour le cas uni-dimensionnel, nous avons finalement choisi de développer notre propre plate-forme parallèle plus adaptée à notre cas propre, plus rapide et plus portable sur les différentes architectures que nous utilisons.

9.2 Implémentation parallèle du modèle bi-dimensionnel

Dans le but d'optimiser le temps d'exécution de notre simulation, nous avons envisagé d'utiliser une méthode de décomposition de domaine régulière pour le portage de notre code sur une plate-forme parallèle à mémoire distribuée.

D'un point de vue pratique, nous avons décidé de ne pas réutiliser la bibliothèque d'échanges de messages PVM ([GEIST et al., 1994]) comme pour la mise en œuvre informatique de la simulation uni-dimensionnelle. En effet, la version 3.4 de cette bibliothèque sortie en 1997 (mais qui n'est toujours pas installée sur le CRAY T3E et l'ORIGIN 2000 que nous avons utilisés, au mois de mars 1999) n'offre pas plus que les versions antérieures, de fonctionnalités de construction de topologies de processus ou de construction de types dérivés. Par contre, la bibliothèque MPI ([SNIR et al., 1995]) vers laquelle nous nous sommes tournés, offre, elle, plusieurs primitives qui permettent notamment de créer une grille virtuelle de processus, de déterminer les processus voisins...ce qui simplifie notablement la programmation [BRUGEAS, 1996].

9.2.1 La topologie cartésienne de processus

Pour toutes les communications dites "locales", parce qu'elles ne concernent que des échanges de données aux interfaces entre les sous-domaines, il est intéressant de disposer les processus suivant une topologie régulière. MPI, contrairement à PVM, offre la possibilité de définir des topologies virtuelles de type cartésien. Cette possibilité s'avère très intéressante dans la mesure où elle constitue un moyen pratique de dénomination des processus qui correspond, en plus, au schéma de communication et permet donc à MPI d'optimiser les communications.

Il faut tout d'abord rappeler qu'une topologie virtuelle de processus en MPI est une notion qui ne dépend, pour sa description, que de l'application qui la crée. Elle est indépendante de l'architecture de la machine cible et donc très portable.

Au cours d'une simulation bi-dimensionnelle telle que nous l'avons mis en place, il se produit deux types de tâches bien distincts. D'une part l'itération en temps de l'automate cellulaire, et d'autre part la collecte des résultats (c'est-à-dire la collecte des portions d'images de la zone de subduction obtenues après calcul). Pour mettre en œuvre cette division des tâches sans désynchroniser l'ensemble des processus, nous avons décidé de partitionner cet ensemble en deux catégories. Ainsi, nous avons d'une part un groupe (ou plus exactement un communicateur [1], pour reprendre la terminologie de

1. D'après [SNIR et al., 1995], un communicateur est une "boite noire" dotée d'un certain nombre d'attributs et de règles simples concernant sa création, son utilisation et

MPI) "d'itérateurs", qui se charge de mettre à jour l'automate cellulaire à chaque pas de temps. Et, d'autre part, nous avons un communicateur de "superviseurs" chargés de la collecte des résultats et de leur stockage sur disque. Dans la pratique ce second groupe de processus se réduit à un seul élément. Cette division des tâches est justifiée par le fait que les travaux correspondants ainsi que les communications entre processus qui leurs sont associés, sont bien différents par nature. Ainsi, alors que les "itérateurs" ne communiquent qu'avec leurs voisins immédiats, le "superviseur", lui, communique avec tous les "itérateurs".

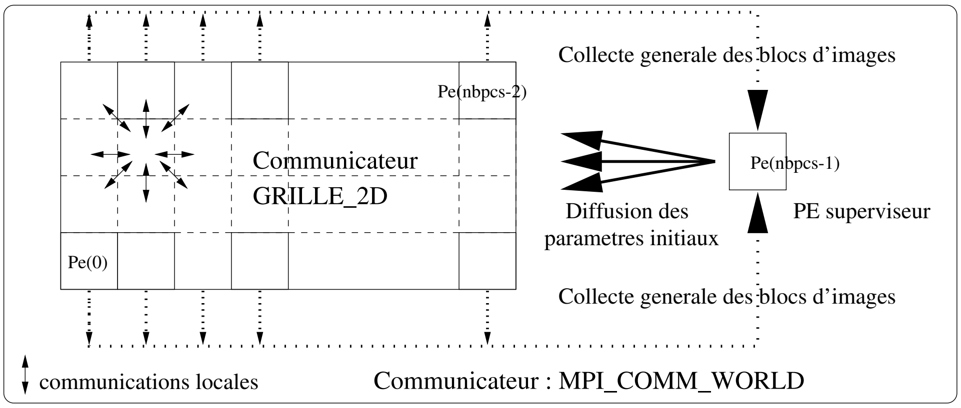

FIG. 9.1 – *Décomposition de domaine avec* MPI

En ce qui concerne la décomposition de domaine elle-même, étant donné que le système discret a une structure très régulière, nous avons distribué les données uniformément à l'ensemble des processus du communicateur des "itérateurs". Ainsi, à chacun des itérateurs est affecté un ensemble de cellules contiguës qui représentent une portion d'image de la zone de subduction à une date donnée.

Pour notre simulation, nous avons partitionné le communicateur global MPI_COMM_WORLD en deux sous-communicateurs (fig. 9.1) avec la primitive MPI_comm_split() : le premier constitué du seul "superviseur" et le second constitué du reste des processeurs du communicateur global. Au cours de la simulation, alors que le superviseur est chargé de gérer les accès aux disques et les diverses entrées-sorties et de collecter les portions d'images, les autres processeurs sont, eux, chargés d'itérer l'automate cellulaire en pas de temps. Ces "itérateurs", parce qu'ils doivent échanger leurs données aux frontières avec leurs processeurs voisins, vont être disposés sur une topologie cartésienne virtuelle régulière à deux dimensions appelée GRILLE_2D. En ef-

sa destruction. Un communicateur permet de définir un domaine de communication au sein de l'ensemble des processus, qui pourra être utilisé pour toutes sortes de communications locales.

fet, ces communications locales se renouvelant excessivement fréquemment, justifient la création d'un support de communication spécial (un communicateur, donc). Nous créons cette topologie avec la primitive MPI_Cart_create() et offrons à l'utilisateur de la simulation, la possibilité d'imposer lui-même, dynamiquement, au lancement de l'exécutable, la taille de la grille des processus itérateurs dans chacune des dimensions.

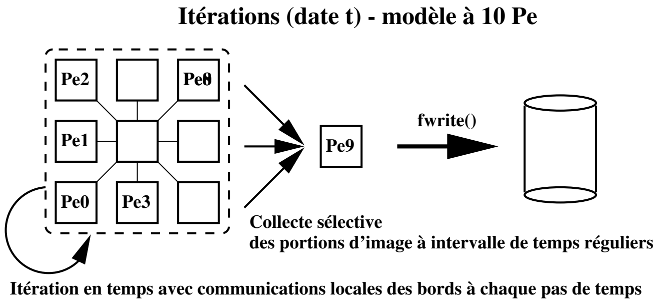

FIG. 9.2 – *Partitionnement de l'ensemble des processus en deux communicateurs*

Les primitives MPI_Cart_coords(), MPI_Cart_shift() et MPI_Cart_rank() de la bibliothèque permettent d'obtenir les coordonnées des processeurs (et de leurs voisins) à partir du rang du processeur courant et inversement.

9.2.2 Les types dérivés pour les communications entre sous-domaines

L'envoi de message est une opération coûteuse, à cause du temps de latence. Il faut donc, pour améliorer les performances de la simulation, s'efforcer d'envoyer toutes les cellules aux frontières, à un même processus destinataire en une seule fois.

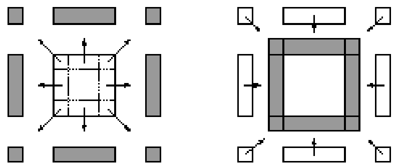

FIG. 9.3 – *Echange de messages entre un itérateur et ses voisins*

La bibliothèque d'échanges de messages MPI propose trois mécanismes différents de regroupement de données individuelles au sein d'un même message. L'un d'eux consiste en la création de nouveaux types de données dérivés. En effet, dans les communications point à point de notre code, les données échangées sont bien souvent de types plus élaborés que les types "simples" classiques. Ainsi, nos processus de calcul échangent à chaque pas de temps deux lignes de cellules avec leurs voisins N et S, deux colonnes de cellules avec leurs voisins O et E, et des "angles" de quatre cellules avec leurs voisins NO, NE, SO, SE (fig. 9.3).

En effet, pour pouvoir appliquer la fonction de transition locale à chacune des cellules du sous-domaine courant, il faut que celles-ci aient connaissance des états respectifs des 24 cellules voisines. Or, aux frontières d'un sous-domaine donné, il faut aussi que les cellules aient connaissance des états de leurs voisines, c'est-à-dire des états de cellules situées théoriquement sur le processus voisin. Pour ce faire, il faut donc qu'à chaque itération, les processus voisins se communiquent les états des cellules situées à leurs interfaces respectives.

Avec une bibliothèque d'échanges de messages classique (comme PVM), l'envoi et la réception de tels blocs de données auraient nécessité l'écriture d'un certain de nombre de boucles et entraîné, du même coup, des surcoûts de compactage/décompactage et de recopie des contenus de messages. Avec MPI, nous avons créé des types dérivés et obtenu, avec les primitives MPI_Type_struct() et MPI_Type_vector(), respectivement un type structuré appelé Type_cellule et trois types de données homogènes vectoriels à pas constant (_type_lignes, _type_angles, _type_colonnes, voir fig. 9.3 et fig. 9.6).

9.2.3 Les types dérivés pour la collecte des portions d'image

Au bout d'un certain nombre d'itérations de la fonction de transition, il faut sauver la configuration de l'automate afin de pouvoir générer, par post-traitement graphique, une animation de l'évolution de l'automate cellulaire. En fait, il suffit juste d'enregistrer la valeur du champ "couleur" de chacune des cellules. Or, du fait de la décomposition de domaine, les valeurs à sauvegarder sur disque sont distribuées sur l'ensemble des itérateurs. Il faut donc les centraliser sur un même processus pour les stocker sur un même disque.

Pour ce faire, et parce que nous ne disposons pas de la possibilité d'accéder concurremment et directement à un même fichier, nous avons adopté la stratégie suivante : regroupement des données distribuées sur un unique processeur (le superviseur) et écriture sur disque après réordonnancement et reconstitution de chacune des images de la simulation. L'important dans cette étape est de ne désynchroniser aucun itérateur, afin de ne pas déséquilibrer la charge d'un processus itérateur par rapport à ses voisins. C'est pour cette raison, mais aussi parce que les entrées-sorties sur disque sont des opérations coûteuses, que le processus superviseur se charge entièrement de cette collecte des "blocs d'images" (ou tableau de ces champs "couleur" mis bout à bout) et du réordonnancement et de l'écriture consécutifs.

Comme nous pouvons le constater sur la fig. 9.4, nous "inversons verticalement", à la collecte, le tableau des pixels de chaque sous-domaine. En effet, alors que le tableau de cellule est lu de bas en haut et de la gauche vers la droite, ligne par ligne (nous sommes en C), les utilitaires d'affichage lisent, eux, un tableau de pixels de haut en bas et de la gauche vers la droite. C'est aussi pour cette raison que nous "inversons verticalement" la position de chaque bloc de pixels dans l'ensemble de l'image (voir fig. 9.5).

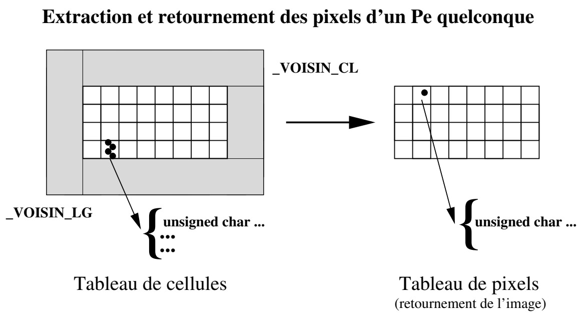

FIG. 9.4 – *Formation du tableau des pixels à partir du tableau de cellules*

Nous avons donc créé le type dérivé Type_bloc_image avec la primitive MPI_Type_indexed() en considérant le tableau des cellules comme un tableau d'octets et en en extrayant l'octet correspondant à la couleur de la cellule avec des décalages à pas non constants (puisqu'il s'agit de ne prendre en compte que les cellules non situées sur les bords, les cellules fantômes n'étant en fait que des cellules redondantes du point de vue de l'information de couleur qui leur est attachée). Pour obtenir des tailles de champs alignées en mémoire, nous avons utilisé la primitive MPI_Type_extent().

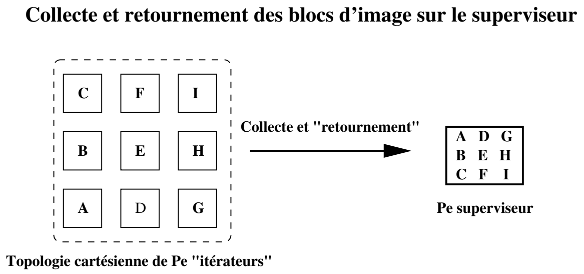

FIG. 9.5 – *Collecte et réordonnancement des portions d'images*

Enfin, après émission d'une donnée "Type_bloc_image" par un itérateur quelconque au cours de la simulation, il faut que le superviseur réceptionne le tableau d'octets correspondant et le stocke au bon endroit dans l'image globale. Pour ce faire, nous avons créé le type dérivé Type_image_globale avec la primitive MPI_Type_vector().

Une fois encore, ce dernier type nous permet de réaliser deux opérations en une seule puisque, avec une librairie "conventionnelle", il nous aurait d'abord fallu réceptionner le bloc d'image avant de le scinder en petits blocs à répartir (ou recopier) dans divers endroits de l'image globale.

9.2.4 Utilisation de "ghost" cellules aux interfaces des sous-domaines

Dans le cadre d'une distribution bi-dimensionnelle (ou même uni-dimensionnelle) des données, chaque sous-domaine doit envoyer ses cellules aux frontières aux processus voisins et, en retour, recevoir leurs propres cellules. Il faut donc qu'à chaque itération, chaque processus stocke des cellules n'appartenant pas à son propre sous-domaine (avant de les utiliser pour mettre à jour les cellules aux bords du sous-domaine). Pour résoudre ce problème de stockage en local d'informations non-locales, nous avons introduit dans le modèle une zone de "recouvrement" de "ghost" cellules. Ces cellules qui n'ont pas "d'existence réelle" vis-à-vis de la modélisation sont physiquement stockées sur le pourtour du sous-domaine de cellules du processus courant. Elles étendent ainsi de deux rangées ou colonnes les cellules qui sont situés sur les "bords intérieurs" (voir fig. 9.6).

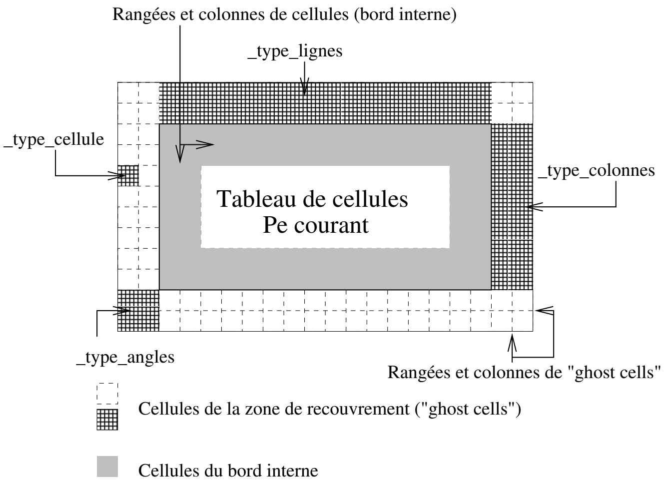

FIG. 9.6 – *Les cellules fantômes aux frontières d'un sous-domaine*

Ainsi, à chaque pas de temps, les "ghost" cellules du sous-domaine courant prennent pour nouvelles valeurs celles des cellules correspondantes dans les sous-domaines voisins. Par exemple, les deux colonnes de "ghost" cellules du bord droit prennent les valeurs des deux colonnes de cellules du "bord intérieur" gauche envoyées par le processus situé à l'est du processus courant. Tandis que les deux colonnes de cellules du "bord intérieur" droit du sous-domaine du processus courant sont envoyées au processus de l'est afin d'alimenter ses propres deux colonnes de "ghost" cellules situées sur le bord gauche du sous-domaine concerné.

9.3 Algorithme simplifié

Comme nous le montrons dans l'algorithme suivant, chaque processus itérateur boucle un nombre fini de fois (tant qu'une certaine condition de sauvegarde (1) n'est pas réalisée) sur la production du "bloc image" correspondant à l'état des cellules de son sous-domaine, après application de la fonction de transition. Si, au cours d'une de ces itérations, une condition d'envoi (2) est satisfaite, alors il envoie son "bloc image" au processus superviseur. De son côté, le superviseur sert chaque itérateur tour à tour, en stockant chacun des "blocs images" envoyés au bon endroit. Les deux conditions de sauvegarde et d'envoi précitées sont fonctions de trois paramètres donnés au lancement de la simulation : nombre total d'images à produire, période entre deux images consécutives et numéro de la première image. Une condition importante de la simulation est que le nombre de cellules de l'automate soit un multiple du nombre de processus itérateurs dans chaque dimension !

Dans le but d'améliorer les performances de la simulation parallèle, nous avons fait en sorte qu'il y ait un recouvrement des temps de communication et des temps de calcul dans la portion de code correspondant à l'application de la fonction de transition globale de l'automate. Ainsi, chaque itérateur effectue un envoi non-bloquant des cellules du "bord interne" et enchaîne immédiatement (sans se soucier de leurs réception par les itérateurs voisins) sur la mise à jour des cellules de son sous-domaine propre qui ne nécessitent pas la connaissance d'état de cellules mises à jour au niveau des sous-domaines voisins. A la réception bloquante des cellules de l'interface en provenance des 8 sous-domaines voisins, l'itérateur termine la mise à jour des cellules situées au bord de son sous-domaine.

L'algorithme parallèle

```
Initialisation de MPI
Création de la topologie cartésienne de processus
Création des types de données dérivés
Si le processus courant est superviseur alors:
   Itérer
      Pour chaque itérateur faire
         Identifier l'itérateur à l'origine du nouveau message,
         Réceptionner le bloc image correspondant,
         Et le placer simultanément au bon endroit dans
            l'image globale,
      Fin du pour.
      Sauver l'image globale sur disque au format Sun-Raster,
   Tant que la condition de sauvegarde (1) n'est pas réalisée.
Sinon
   Itérer
      Envoi non bloquant des cellules du "bord interne"
         aux 8 itérateurs voisins,
      Application de la fonction de transition aux cellules
         "propres" (placées à plus de 4 cellules des bords),
      Réception bloquante des "ghost" cellules en provenance
         des itérateurs voisins,
      Application de la fonction de transition aux cellules
         situées sur le bord interne (i.e. aux cellules situées
         à 2 ou 3 cellules des bords),
      Si la condition d'envoi (2) est réalisée alors:
         Envoi du bloc d'image local ("ghost" cellules
            exclues) au superviseur,
      Fin du si.
   Tant que la condition de sauvegarde (1) n'est pas réalisée.
Fin du si.
Synchronisation de l'ensemble des processus,
Sortie de MPI.
```

Quatrième partie

Evaluation des performances, visualisation et interprétations

Chapitre 10

Evaluation des performances

Dans ce dernier chapitre, nous nous attachons à évaluer l'ensemble de notre travail du point de vue de l'optimisation parallèle d'abord, et du point de vue du rendu-visuel ensuite. Comme nous allons le constater à la lecture des résultats que nous y exposons, paralléliser le modèle uni-dimensionnel ne nous permet pas d'optimiser réellement les temps de calcul. Dans le cas bi-dimensionnel, par contre, l'optimisation parallèle donne des résultats très positifs.

10.1 Les performances de la simulation uni-dimensionnelle

Comme nous pouvons le constater au vu du tableau de la fig. 10.1 et des fig. 10.2, 10.3 et 10.4, les performances de la simulation parallèle uni-dimensionnelle s'accomodent mal de l'augmentation du nombre de processeurs. En effet, la quantité de données à traiter sur chaque processeur, du fait de la décomposition de domaine régulière, devient rapidement trop faible par rapport à la quantité d'informations à échanger aux interfaces des sous-domaines, lorsque le nombre de processeurs augmente. Elles restent cependant acceptables pour un faible nombre de processeurs. D'après la fig 10.4, l'efficacité du parallèlisme est très rapidement nulle : les temps de communication ne sont très vite plus recouverts par les temps de calcul... les processeurs passent la totalité de leur temps à attendre les informations aux interfaces de leurs sous-domaines [1].

Contrairement au cas bi-dimensionnel que nous analysons ci-après, il faut donc en conclure que notre choix d'une optimisation parallèle par la méthode de décomposition de domaine ne convient que pour un petit nombre de processeurs.

1. Ce que confirme très bien la commande *top* sur l'ORIGIN 2000 qui prouve que l'activité des processeurs est quasi nulle.

Nombre de processeurs	CRAY T3E			ORIGIN 2000		
	Temps	Accélération	Efficacité	Temps	Accélération	Efficacité
1	36.77	1.00	1.000	29.33	1.00	1.000
2	25.09	1.47	0.733	23.76	1.23	0.617
4	15.26	2.41	0.602	21.25	1.38	0.345
5	12.39	2.97	0.594	20.89	1.40	0.281
8	10.93	3.36	0.421	24.12	1.22	0.152
10	8.32	4.42	0.442	27.12	1.08	0.108
20	8.88	4.14	0.207	31.34	0.94	0.047
25	7.85	4.69	0.187	38.45	0.76	0.031
40	8.55	4.30	0.107			
50	9.04	4.07	0.081			
100	10.31	3.57	0.036			
125	10.32	3.56	0.029			
200	9.62	3.82	0.019			
250	9.94	3.70	0.015			

FIG. 10.1 – *Comparaison des temps de calcul (en secondes), accélération et efficacité de la simulation uni-dimensionnelle, en fonction du nombre de processeurs pour un même ensemble de conditions initiales, entre un* CRAY T3E *à 256 processeurs et une* ORIGIN 2000 *à 32 processeurs*

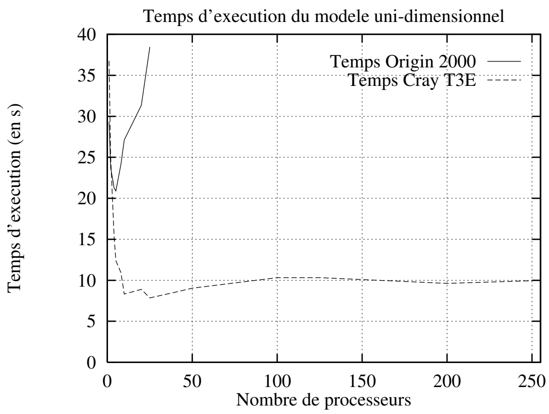

FIG. 10.2 – *Temps de simulations comparés selon la machine cible pour le modèle uni-dimensionnel*

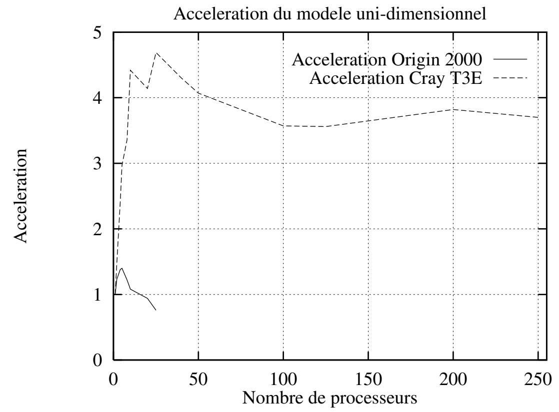

FIG. 10.3 – *Accélérations comparées selon la machine cible pour le modèle uni-dimensionnel*

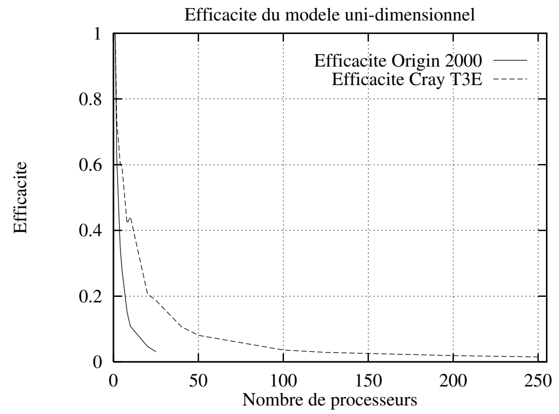

FIG. 10.4 – *Efficacités comparées selon la machine cible pour le modèle uni-dimensionnel*

10.2 Les performances de la simulation bi-dimensionnelle

10.2.1 Choix de la meilleure topologie pour un nombre de processus donné

Parce que nous avons choisi de faire appel aux types de données dérivés et aux topologies de processus afin d'optimiser au maximum l'écriture de la simulation bi-dimensionnelle, nous devons découper le tableau global de l'ensemble des cellules, de façon régulière, en plusieurs sous-domaines identiques. En effet, il importe que les types de données homogènes vectoriels à pas constant _type_lignes, _type_angles, _type_colonnes (voir fig. 9.6) aient la même signification pour tous les itérateurs. Cette condition implique donc que la longueur et la hauteur commune à chacun de ces sous-domaines soient des diviseurs de la longueur et de la hauteur (respectivement) du tableau global de l'ensemble des cellules. C'est pour cette raison que nous regroupons dans le tableau de la fig. 10.5 l'ensemble des topologies acceptables pour un nombre de processeurs donnés.

Dans ce même tableau, pour une topologie de processus $nbpe_{ligne} \times nbpe_{colonne}$ donnée, nous proposons également la valeur du ratio surface des interfaces/surface globale correspondante. C'est-à-dire la valeur du pourcentage suivant (dans le cas d'un tableau global de 1000 par 200 cellules) :

$$\frac{4 \times (4 + \frac{1000}{nbpe_{ligne}} + \frac{200}{nbpe_{colonne}})}{(\frac{1000}{nbpe_{ligne}} + 4) \times (\frac{200}{nbpe_{colonne}} + 4)} \times 100$$

Pour minimiser le temps passé à échanger des données aux bords par rapport au temps passé à appliquer la fonction de transition locale à l'ensemble des cellules d'un sous-domaine donné, il s'impose donc de minimiser ce ratio. C'est ce que nous avons fait pour établir les temps de calcul donnés dans le tableau de la fig.10.6

10.2.2 Efficacité de la parallélisation de la simulation bi-dimensionnelle

Comme nous pouvons le constater à la lecture du tableau de la fig. 10.6 et de la fig. 10.7, à nombre de processeurs de calcul équivalent, l'ORIGIN 2000[2] est plus rapide que le CRAY T3E. Par contre, la fig. 10.8 montre que l'accélération est plus forte sur le CRAY T3E que sur l'ORIGIN 2000. Cette particularité est probablement explicable par le fait que le ratio temps de communication sur temps de calcul est plus favorable au CRAY T3E (qui calcule 2 fois moins vite que l'ORIGIN 2000). En effet, en ce qui concerne les

2. Lorsque l'on utilise les options de compilation suivantes : -r10000 -mips4 -n32 -Ofast=ip27 -OPT:IEEE_arithmetic=3.

nb proc	Topologies associées (nb de processeurs en ligne * nb de processeurs en colonnes) Ratios associés (rapport du nb de cellules aux interfaces sur le nb total de cellules d'un sous-domaine)											
1	1 * 1 **2,35**											
2	1 * 2 4,23	2 * 1 **2,74**										
4	1 * 4 7,78	2 * 2 4,61	4 * 1 **3,50**									
5	1 * 5 9,45	5 * 1 **3,88**										
8	1 * 8 14,14	2 * 4 8,14	4 * 2 5,36	8 * 1 **5,00**								
10	1 * 10 17,00	2 * 5 9,81	5 * 2 **5,73**	10 * 1 **5,73**								
16	2 * 8 14,48	4 * 4 8,87	8 * 2 **6,83**									
20	1 * 20 28,86	2 * 10 17,33	4 * 5 10,52	5 * 4 9,22	10 * 2 **7,54**	20 * 1 9,22						
25	1 * 25 33,60	5 * 5 **10,87**	25 * 1 **10,87**									
32	4 * 8 15,15	8 * 4 **10,28**										
40	1 * 40 44,67	2 * 20 29,14	4 * 10 17,98	5 * 8 15,48	8 * 5 11,91	10 * 4 **10,97**	20 * 2 **10,97**	40 * 1 15,48				
50	1 * 50 50,20	2 * 25 33,86	5 * 10 18,30	10 * 5 **12,59**	25 * 2 **12,59**	50 * 1 18,30						
64	8 * 8 **16,47**											
80	2 * 40 44,89	4 * 20 29,70	8 * 10 19,25	10 * 8 17,11	20 * 4 **14,27**	40 * 2 17,11						
100	1*100 66,80	2 * 50 50,40	4 * 25 34,38	5 * 20 29,97	10*10 19,87	20 * 5 **15,82**	25 * 4 **15,82**	50 * 2 19,87	100*1 29,97			
125	5 * 25 34,64	25 * 5 **17,36**	125*1 34,64									
160	4 * 40 45,32	8 * 20 30,79	20 * 8 **20,18**	40 * 4 **20,18**								
200	1*200 80,08	2*100 66,93	4 * 50 50,79	5 * 40 45,53	8 * 25 35,40	10*20 31,32	20*10 22,84	25 * 8 **21,63**	40 * 5 **21,63**	50 * 4 22,84	100*2 31,32	200*1 45,53
250	5 * 50 50,98	10*25 35,90	25*10 **24,24**	50 * 5 **24,24**	125*2 35,90	250*1 50,98						

FIG. 10.5 – *Inventaire des topologies de processus acceptables pour un réseau d'automates cellulaires de 1000 par 200 cellules. Les ratios écrits en gras (qui correspondent au minimum des valeurs de rapports de la taille des cellules sur les interfaces d'un sous-domaine par rapport au total des cellules d'un sous-domaine) indiquent la topologie qu'il faut choisir, pour un nombre de processeurs donné afin, d'obtenir la meilleure efficacité de parallélisme*

Nombre de processeurs	CRAY T3E			ORIGIN 2000		
	Temps	Accélération	Efficacité	Temps	Accélération	Efficacité
1	958.90	1.00	1.000	469.70	1.00	1.000
2	960.30	1.00	0.499	474.80	0.99	0.495
3	521.10	1.84	0.613	273.40	1.72	0.573
5	274.30	3.50	0.699	143.75	3.27	0.653
6	219.69	4.36	0.727	117.91	3.98	0.664
9	149.16	6.43	0.714	81.23	5.78	0.642
11	116.95	8.20	0.745	66.05	7.11	0.646
17	77.78	12.33	0.725	44.19	10.63	0.625
21	64.29	14.92	0.710	37.10	12.66	0.603
26	58.17	16.48	0.634	34.36	13.67	0.526
33	43.14	22.23	0.674			
41	36.88	26.00	0.634			
51	30.87	31.06	0.609			
65	28.36	33.81	0.520			
81	17.62	54.42	0.672			
101	15.04	63.76	0.631			
126	13.40	71.56	0.568			
161	10.97	87.41	0.543			
201	9.64	99.47	0.495			
251	8.62	111.24	0.443			

FIG. 10.6 – *Comparaison des temps de calcul (en secondes), accélération et efficacité de la simulation bi-dimensionnelle, en fonction du nombre de processeurs pour un même ensemble de conditions initiales, entre un* CRAY T3E *à 256 processeurs et une* ORIGIN 2000 *à 32 processeurs*

temps de communications respectifs, nous pouvons les supposer relativement proches, puisque les latences moyennes des deux réseaux de communication sont proches ($1\mu s$ pour le CRAY T3E et 773 ns pour l'ORIGIN 2000) ainsi que leurs bandes passantes[3].

Par ailleurs, en observant l'accélération de la simulation sur le CRAY T3E en fig. 10.8, on note une réelle chute de l'efficacité de l'optimisation parallèle pour 65 processeurs. En effet, alors qu'elle est supérieure à six dixième pour 51 et 81 processeurs, elle tombe brutalement à la valeur $0,52$. Nous supposons que ceci est dû au fait que plus la taille des sous domaines est grande proportionnellement à la taille de leurs interfaces, meilleure est l'efficacité du code parallèle. En d'autre termes, à taille de problème globale constante[4] l'efficacité décroît quand le sous domaine diminue et que la proportion de cellules aux interfaces (et donc à échanger) augmente. Or, dans le cas à 65 processeurs, nous sommes obligés de travailler sur une grille de 8×8 processeurs itérateurs (en effet 16 n'est pas un diviseur de 1000) ce qui implique

3. Ce qui confirme d'ailleurs que l'accélération et l'efficacité ne sont pas toujours de bons critères pour comparer des architectures...

4. C'est notre cas ici, puisque ces mesures ont été faites dans les mêmes conditions initiales et ont générées chacunes 20 images par pas de 100 unités de temps.

que chaque itérateur doit traiter un sous domaine de 129×29 cellules, avec 16,47% de celles-ci situées aux interfaces. Dans le cas à 81 processeurs par contre, nous travaillons sur une grille de 20×4 processus itérateurs ce qui implique que chaque itérateur doit traiter un sous domaine de 54×54 cellules, avec 14,27% de celles-ci situées aux interfaces. Cette chute des performances pour 65 processus est donc explicable.

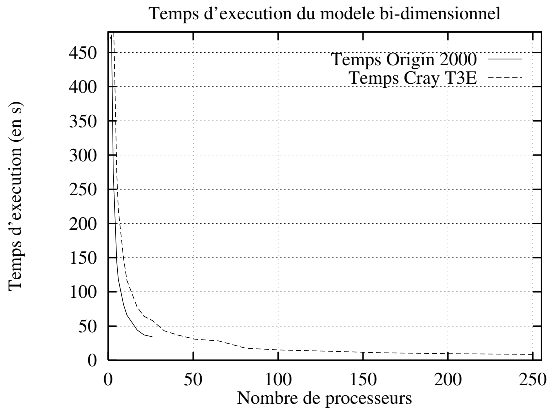

FIG. 10.7 – *Temps de simulations comparés selon la machine cible pour le modèle bi-dimensionnel*

Nous pouvons par conséquent en conclure qu'il importe aussi de disposer la grille des processus de façon adéquate (afin de diminuer la taille des problèmes aux interfaces) si l'on souhaite obtenir un rendement optimal.

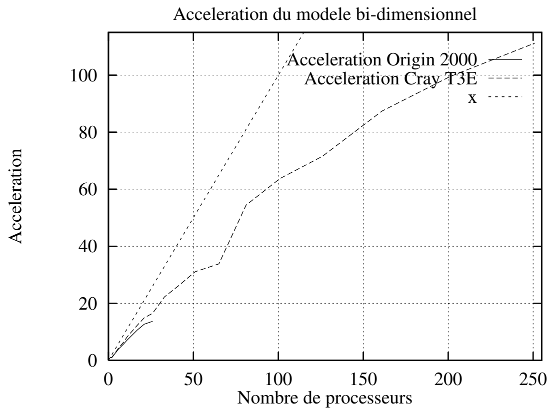

FIG. 10.8 – *Accélérations comparées selon la machine cible pour le modèle bi-dimensionnel*

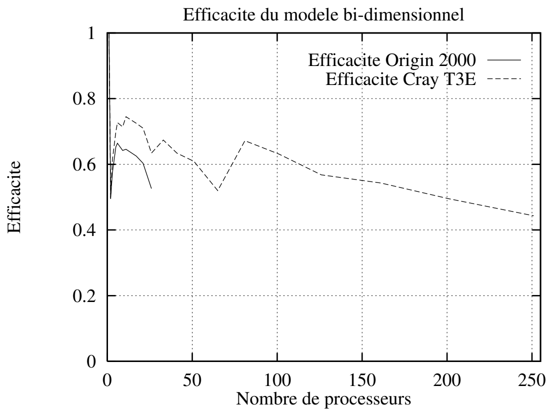

FIG. 10.9 – *Efficacités comparées selon la machine cible pour le modèle bi-dimensionnel*

L'accélération de cette simulation bi-dimensionnelle est quasi-linéaire et croissante sur l'intervalle d'étude (fig. 10.8). L'intérêt d'une solution parallèle est flagrant puisque, alors que le temps de simulation séquentiel est de plus d'un quart d'heure dans le cas (relativement restreint) que nous avons exposé précédemment, il descend à 8,62 secondes dans le cas parallèle sur 251 processeurs du Cray t3e ! L'utilisation des nombreuses fonctionnalités de la bibliothèque de communication mpi pour les décompositions de domaine s'est donc avérée rentable dans le cas de notre simulation bi-dimensionnelle.

Chapitre 11

Rendus visuels comparés des deux simulations

11.1 Les moyens utilisés pour la visualisation

11.1.1 Le serveur graphique de l'IDRIS

Le serveur graphique est un Power Challenge XL de Silicon Graphics, machine multi-processeurs à mémoire partagée (SMP). Il est doté de 4 processeurs R10000 possédant une fréquence d'horloge de 195 MHz, de 2 Go de mémoire globale et de 200 Go d'espace disque. Cette configuration offre une puissance crête globale de 1,6 Gflops et 2400 Mips. Il peut travailler en simple précision 32 bits IEEE ou en double précision 64 bits IEEE.

Son système d'exploitation est un IRIX 6.2. La gestion de la répartition de ses ressources se fait par NQS[1].

11.1.2 Les logiciels ImageMagick et Mpeg player

ImageMagick, version 3.7.2, est un ensemble d'utilitaires dédié à l'affichage ou la manipulation d'images en environnement X Window. Il est écrit en langage C ANSI, téléchargeable à l'adresse

ftp://ftp.wizards.dupont.com/pub/ImageMagick/

et potentiellement compilable sur toute machine Unix (Linux compris) et VMS. Dans l'ensemble des utilitaires de ce logiciel, nous avons principalement utilisé l'outil *convert* qui nous a permis de transformer un ensemble de clichés du format Sun-Raster ou Raw (RGB) vers le format d'animation MPEG et les outils *display* et *animate* pour visualiser respectivement des images fixes et des animations. En fait, pour lire les fichiers au format MPEG, nous avons préféré utiliser le logiciel de visualisation du domaine public : *Mpeg player*.

1. Network Queuing System.

11.2 Visualisation de la simulation uni-dimensionnelle

Sur les quelques copies d'écran - résultat d'une exécution de la simulation uni-dimensionnelle - présentées ici, nous retrouvons l'ensemble des couches tel que décrit dans la section 5.2.1 de ce document. Ainsi se localise, de haut en bas, sur chacune des coupes : une portion d'océan (en mauve) par 6000 mètres de profondeur, une portion de plaque continentale subdivisée en quatre couches (en gris, mauve, vert-foncé et jaune, le jaune symbolisant ici la couche de matériel continental fortement hydrofracturé susceptible d'être décroché de la base de la plaque chevauchante et charrié vers les profondeurs), une zone interstitielle intermédiaire entre les deux plaques lithosphériques représentant le matériel de la fosse océanique et du chenal de subduction (en vert clair), une couche représentant le matériel décroché de la marge chevauchante et déposé à la surface de la plaque océanique (en jaune) et, enfin, la plaque océanique elle-même (en marron) avec ses effets de "brisure".

Bien que le découpage du plan de subduction en un réseau d'automates cellulaires soit assez "grossier" dans le cas uni-dimensionnel, on observe tout de même que le rendu visuel à l'exécution est relativement satisfaisant. Pour ce faire, il a d'abord fallut adapter "empiriquement" quelques paramètres de la règle de transition. Sur les plans de coupe ci-après, on note que l'altitude du point de contact de la subduction au front de la plaque chevauchante reste constante, le pendage du plan de subduction est constant et le front de la marge continentale "recule" dans une proportion (relativement) cohérente avec l'avancée de la plaque plongeante.

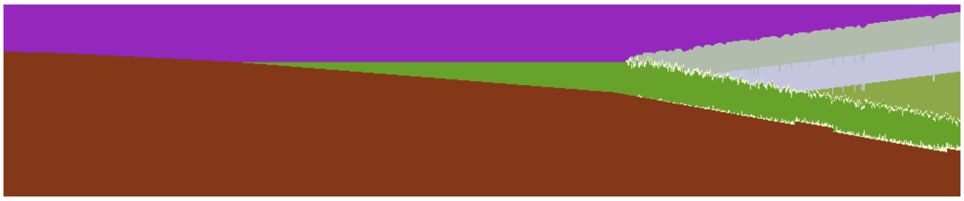

FIG. 11.1 – *Rendu visuel de la simulation uni-dimensionnelle après 1100 pas de temps*

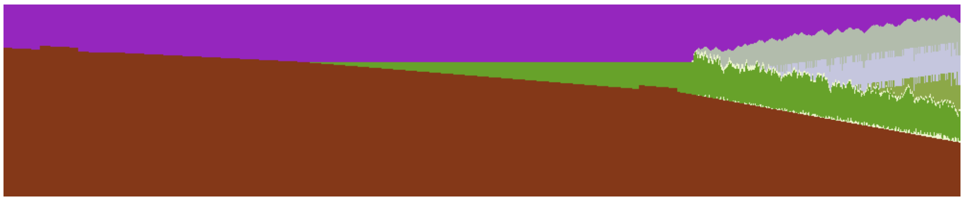

FIG. 11.2 – *Rendu visuel de la simulation uni-dimensionnelle après 3100 pas de temps*

FIG. 11.3 – *Rendu visuel de la simulation uni-dimensionnelle après 5100 pas de temps*

FIG. 11.4 – *Rendu visuel de la simulation uni-dimensionnelle après 7100 pas de temps*

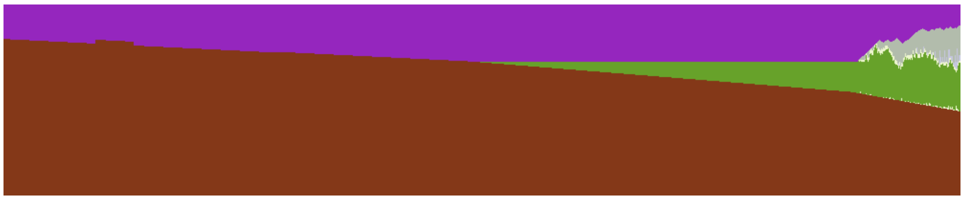

FIG. 11.5 – *Rendu visuel de la simulation uni-dimensionnelle après 9100 pas de temps*

Sur l'ensemble des clichés présentés ici et, de façon plus générale, sur l'ensemble d'une simulation, nous pouvons remarquer que la largeur et la profondeur de la fosse océanique ne varient pas excessivement. Ainsi, à son embouchure, le chenal de subduction est toujours approximativement de 800 mètres de haut, tandis que la largeur de la fosse est toujours approximativement de 10000 mètres.

11.3 Visualisation de la simulation bi-dimensionnelle

De la même façon que dans le cas uni-dimensionnel, le rendu visuel obtenu est satisfaisant. Toutefois, il faut noter qu'ici les résultats sont plus "affinés" puisque l'on a pu singulariser plus facilement chacune des cellules du modèle et lui associer un attribut de couleur. De cette façon, les mouvements des cellules au sein du chenal de subduction apparaissent plus clairement, de même que l'érosion des cellules situées à la base de la plaque chevauchante. Pour "étudier" les effondrements de terrain au sein de la plaque continentale, nous y avons placé des repères (c'est-à-dire que nous avons changé la valeur de l'attribut de couleur des cellules concernées sans pour autant les dénaturer). Ces repères nous permettent de retrouver un phénomène de "failles normales" se produisant au sein de la plaque continentale. Il semblerait aussi que l'ensemble de la simulation rende bien compte des glissements sous-marins. De façon générale, nous avons repris, pour la simulation bi-dimensionnelle, les mêmes codes de couleur que dans le cas uni-dimensionnel. Nous avons cependant choisi de représenter la couche interstitielle à l'aide de deux couleurs (vert clair et marron) afin de mieux visualiser les mouvements en son sein.

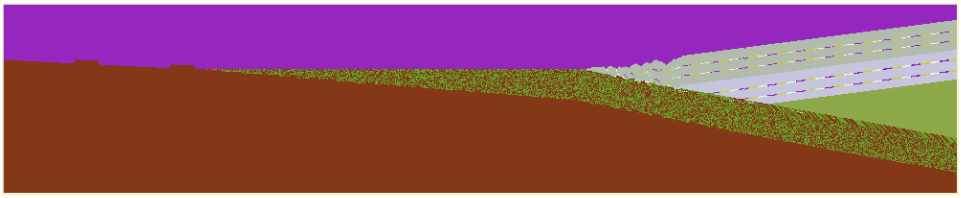

FIG. 11.6 – *Rendu visuel de la simulation bi-dimensionnelle après 1100 pas de temps*

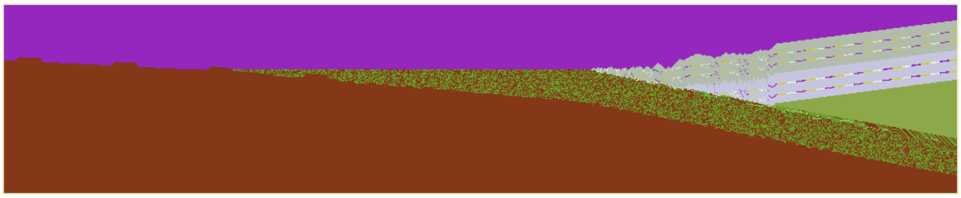

FIG. 11.7 – *Rendu visuel de la simulation bi-dimensionnelle après 3100 pas de temps*

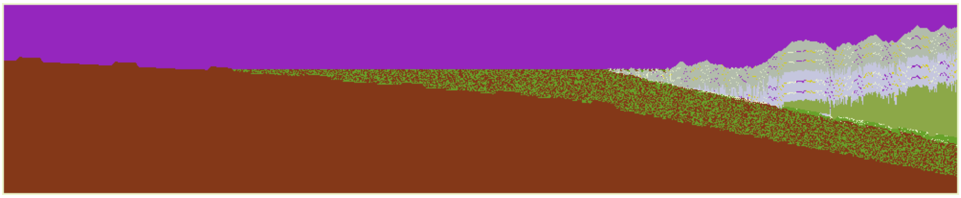

FIG. 11.8 – *Rendu visuel de la simulation bi-dimensionnelle après 5100 pas de temps*

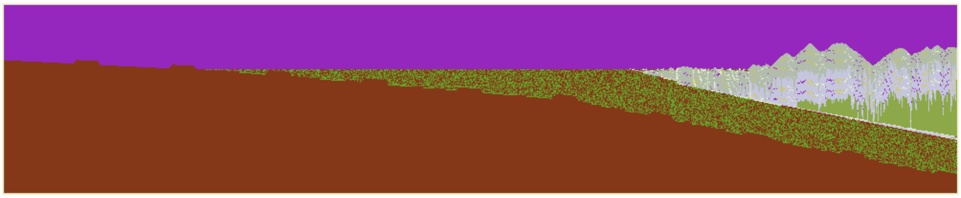

FIG. 11.9 – *Rendu visuel de la simulation bi-dimensionnelle après 7100 pas de temps*

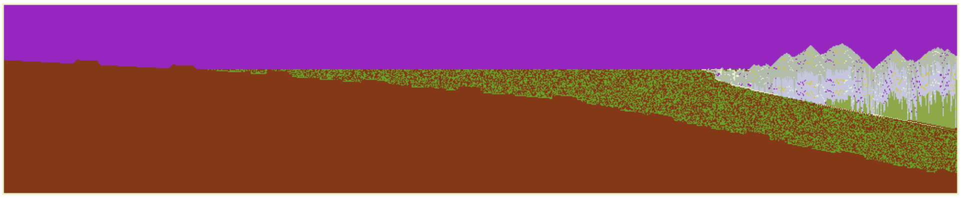

FIG. 11.10 – *Rendu visuel de la simulation bi-dimensionnelle après 9100 pas de temps*

Conclusion et perspectives

L'objectif de cette thèse est de développer une simulation parallèle du phénomène de subduction-érosion en tectonique des plaques, à partir d'une technique par réseau d'automates cellulaires. En effet, comme nous l'expliquons au cours de la première partie de notre travail, les simulations analogiques en "bac à sable" et les simulations analytiques et numériques "globales" de ce processus géotectonique ne donnent pas entière satisfaction. Le but sous-jacent à toutes ces études est clair : il s'agit de fournir une estimation suffisamment précise et détaillée du volume de matière absorbé en subduction, dans le but de procéder à une évaluation des potentialités d'utilisation des grandes fosses océaniques pour éliminer définitivement les déchets radioactifs [Bourgois, 1996].

L'ensemble de notre étude peut-être décomposée en trois étapes majeures. Ainsi, la première d'entre-elles a consisté en une étude générale pour appréhender le contexte du phénomène de subduction-érosion en tectonique des plaques et pour cerner, de la façon la plus exhaustive possible, l'ensemble des mécanismes à mettre en jeu pour les reproduire au niveau d'un modèle numérique. Il importe de savoir, à ce stade de la lecture de ce manuscrit, que cette description n'a pas été arrêtée d'emblée et qu'il y a eu un certain nombre "d'allers-retours" entre cette phase, que nous pouvons qualifier comme étant celle de définition des besoins, et les deux phases qui lui ont succédé.

Dans une seconde étape nous avons cherché à modéliser le phénomène étudié par une technique de réseau d'automates cellulaires. Cette stratégie de modélisation a été choisie au commencement même de notre travail doctoral. Ce développement s'est fait de façon très itérative. Il nous a ainsi fallut procéder par affinages successifs (accompagnés à chaque fois d'une implémentation puis une visualisation des résultats) pour mettre au point une règle de transition au rendu visuel "acceptable". C'est dans le cadre de cette phase de modélisation que nous avons choisi de faire une incursion par un modèle uni-dimensionnel à la suite d'une discussion avec Jérôme Olivier Durand-Lose. Ce "détour" nous a permis de nous poser, dans le cadre restreint de la 1D, un certain nombre de problèmes, comme la représentation du phénomène d'hydrofracturation, et de développer des solutions que nous avons pu appliquer, dans le principe, au modèle bi-dimensionnel.

Dans la troisième et dernière étape de ce travail, nous avons développé deux optimisations parallèles de nos simulations. Si le fait de paralléliser le modèle uni-dimensionnel ne nous permet pas d'optimiser réellement les temps de calcul, dans le cas bi-dimensionnel, par contre, l'optimisation parallèle donne des résultats très positifs. En effet, dans le cas uni-dimensionnel, la quantité de données à traiter sur chaque processeur, du fait de la décomposition de domaine régulière, est trop faible par rapport à la quantité d'informations à échanger aux interfaces des sous-domaines. Dans le cas bi-dimensionnel par contre, l'intérêt d'une solution parallèle est flagrant puisque, alors que le temps de simulation séquentiel est de plus d'un quart d'heure dans le cas (relativement restreint) que nous avons exposé au cours du chapitre précédent, il descend à 8,62 secondes dans le cas parallèle sur 251 processeurs du Cray t3e ! L'utilisation des nombreuses fonctionnalités de la bibliothèque de communication mpi pour les décompositions de domaine s'est donc avérée rentable dans le cas de notre simulation bi-dimensionnelle. Ces simulations ont été portées sur le Cray t3e de l'IDRIS et sur l'Origin 2000 du Pôle Parallélisme Ile-de-France Sud et nous avons pu constater par ailleurs que, pour ce type d'application, l'Origin 2000 est plus rapide que le Cray t3e (à nombre de processeurs de calcul équivalent), mais que l'accélération est plus forte sur le Cray t3e que sur l'Origin 2000.

Au terme de ce travail, nous estimons que nos contributions personnelles ont essentiellement porté sur la modélisation par réseau d'automates cellulaires du phénomène géotectonique étudié et sur la technique de parallélisation d'un tel réseau par décomposition de domaine avec la bibliothèque d'échanges de messages mpi. Cette collaboration interdisciplinaire a permis de réaliser deux "simulateurs" à valeur essentiellement didactique pour l'illustration du phénomène géotectonique de subduction-érosion.

Enfin, pour conclure sur ce point, nous estimons que si la simulation numérique par réseau d'automates cellulaires en géotectonique en est encore à ses balbutiements, l'apparition de modèles en 3 dimensions, fiables, avec des temps de simulation suffisamment courts, lui donnera ses "lettres de noblesse" et la rendra utilisable comme outil de prédiction et d'aide à la décision.

En ce qui concerne les développements qui pourraient être faits à l'issue de cette thèse, nous pensons qu'il faudrait terminer l'écriture de la plate-forme logicielle spécifique $\mathcal{LAC}$ (Langage pour Automates Cellulaires) ce qui pourrait simplifier le travail d'un expérimentateur futur. Cette plate-forme, entamée avec l'aide de Vincent Lefebvre comporterait à terme : un langage propre dédié à la programmation des automates cellulaires bi-dimensionnels, un petit compilateur associé[2] qui génèrerait du code en langage C avec des

2. Écrit avec Lex et Yacc.

appels aux primitives de la bibliothèque d'échange de messages MPI (pour le parallélisme) et une interface graphique[3] pour l'initialisation de l'automate cellulaire.

En ce qui concerne la collaboration interdisciplinaire et l'utilisation de ce travail pour la géotectonique, nous estimons qu'une "analyse du comportement" des simulations, accompagnée d'une qualification de la concentration des déformations au sein de la plaque chevauchante pourrait être intéressante.

3. Ecrite en Perl/Tk.

Bibliographie

ACKLAM, E. and LANGTANGEN, H. P. (1998). Parallelization of explicit finite difference schemes via domain decomposition. Technical report, Oslo Scientific Computing Archive. http://www.math.uio.no/OSCA.

AHRENS, T. J., editor (1995). *Global Earth Physics : A Handbook of Physical Constants.* AGU Reference Shelf 1. American Geophysical Union.

ALMASI, G. S. and GOTTLIEB, A. (1994). *Highly parallel computing.* The Benjamin/Cummings Pub. Company, Inc.

ARNOLD, A., BEAUQUIER, J., BÉRARD, B., and ROZOY, B. (1992). *Programmes parallèles : modèles et validation.* Armand Colin.

AUBOUIN, J., BOURGOIS, J., and AZÉMA, J. (1984). A new type of active margin : the convergent extensional margin, as exemplified by the middle america trench off guatemala. *Earth and Planetary Science Letters*, 67:211–218.

BAK, P., TANG, C., and WIESENFELD, K. (1987). Self-organized criticality : an explanation of 1/f noise. *Phys. Rev. Lett.*, 59(4):381–384.

BOILLOT, G. (1996). *La dynamique de la lithosphère.* Enseignement des Sciences de la Terre. Masson. Glossaire par Jacques BOUTLER.

BOURGOIS, J. (1993). La fosse d'Amérique Centrale : convergence, accrétion, érosion tectonique. *C.R. Acad. Sci., Sér. Gén., Vie Sci.*, 10(4):285–303.

BOURGOIS, J. (1996). Un processus naturel pour éliminer définitivement les déchets nucléaires ultimes. *Réalités Industrielles - Une série des Annales des Mines*, pages 5–12.

BOURRET, P., REGGIA, J., and SAMUELIDES, M. (1991). *Réseaux neuronaux - une approche connexionniste de l'Intelligence Artificielle.* Teknea.

BRUGEAS, I. (1996). Utilisation de MPI en décomposition de domaine. http://www.idris.fr. Publication IDRIS.

CARLSON, R. and UYEDA, T. HILDE. S. (1983). The driving mechanism of plate tectonics : relation to age of the lithosphere at trenches.

CAZOULAT, R. (1996a). *Manuel d'utilisation de Small.* Université de Caen.

CAZOULAT, R. (1996b). *Modélisation et simulation de la dynamique de population d'agents.* Thèse de doctorat, Université de Caen.

CHERGUI, J., DUPAYS, I., GIROU, D., and REQUENA, S. (1997). Message Passing Interface. http://www.idris.fr. Formation IDRIS.

CHERGUI, J. and DUPAYS, J.-M. (1995). PVM sur T3D/T3E — Parallel Virtual Machine. http://www.idris.fr. Formation IDRIS.

CHOPARD, C. (1998). Architecture et technologie des ordinateurs II : architecture des machines parallèles. Université de Genève - http://cuiwww.unige.ch/~chopard/.

CLOOS, M. and SHREVE, R. L. (1996). Shear-zone thickness and the seismicity of chilean- and marianas-type subduction zones. *Geology*, 24(2):107–110.

CONDIE, K. C. (1997). *Plate Tectonics and Crustal Evolution.* Butterworth Heinemann, 4^{th} edition.

COUSOT, P. Concepts avancés de programmation parallèle par messages avec la bibliothèque PVM. Majeur Informatique Fondamentale et Applications - Ecole Polytechnique - http://www.ens.fr/~cousot/.

COUSOT, P. Programmation parallèle répartie en C avec messagerie asynchrone utilisant la bibliothèque PVM. Majeur Informatique Fondamentale et Applications - Ecole Polytechnique - http://www.ens.fr/~cousot/.

CREUTZ, M. (1995). Xtoys : cellular automata on Xwindows. In *Lattice'95.* Physics Department, Brookhaven National Laboratory. http://penguin.phy.bnl.gov/www/xtoys.html.

DELORME, M. and MAZOYER, J. (1998). *Cellular automata : a parallel model*, chapter Computations on cellular automata. Mathematics and its applications. Kluwer.

DERCOURT, J. and PAQUET, J. (1995). *Géologie (objets et méthodes).* Dunod, 9e edition.

DESPREZ, F. (1996a). Cours postgrade - Architectures parallèles - Partie I. http://www.idris.fr. Séminaire de l'IDRIS.

DESPREZ, F. (1996b). Cours postgrade - Architectures parallèles - Partie II. http://www.idris.fr. Séminaire de l'IDRIS.

Desprez, F. (1996c). Programmation d'applications par passage de messages. http://www.idris.fr. Séminaire de l'IDRIS.

Dubois, J. (1995). La tectonique des plaques.

Dubois, J. and Diament, M. (1997). *Géophysique*. Enseignement des Sciences de la Terre. Masson.

Duran, J. (1995). La physique du tas de sable. *Revue du Palais de la Découverte*, 23(224):21–39.

Duran, J. (1997). *Sables, poudres et grains (Introduction à la physique des milieux granulaires)*. Editions Eyrolles.

Durand, B. (1994). *Automates cellulaires : réversibilité et complexité*. Thèse de doctorat, Ecole Normale Supérieure de Lyon.

Durand-Lose, J. O. (1996). *Automates Cellulaires, Automates à Partitions et Tas de Sable*. Thèse de doctorat, Université de Bordeaux I.

Eckart, J. D. (1997). *The Cellular Automata Simulation System : Language Reference Manual*. Radford University. http://www.cs.runet.edu/~dana/ca/cellular.html.

Feautrier, P. (1995). Compiling for massively parallel architectures : a perspective. *Microprogramming and microprocessors*, 41:425–439.

Ferber, J. (1995). *Les Systèmes Multi-Agents - Vers une intelligence collective*. Informatique Intelligence Artificielle. InterEditions.

Flynn, M. J. (1966). Very high speed computing system. *Proceedings IEEE 54*, 12:1901–1909.

Flynn, M. J. (1972). Some computer organizations and their effectiveness. *IEEE Transactions on Computer*, C(21):948–960.

Forsyth, D. and Uyeda, S. (1975). On the relative importance of the driving forces of plate motion. *Geophys. J. R. Soc.*, 43:163–200.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994). *PVM: Parallel Virtual Machine (A Users' Guide and Tutorial for Networked Parallel Computing)*. Scientific and Engineering Computation series. MIT Press. http://www.netlib.org/pvm3/book/pvm-book.html.

Gengler, M., Ubéda, S., and Desprez, F. (1996). *Initiation au parallélisme : concepts, architectures et algorithmes*. Masson.

Girou, D. (1995). Introduction au T3D — aspects matériels et logiciels. http://www.idris.fr. Formation IDRIS.

GROPP, W. (1995). Tutorial on MPI: the Message-Passing Interface. http://www.mcs.anl.gov/mpi/tutorial. Mathematics and Computer Science Division - Argonne National Laboratory - University of Chicago.

GRUMBACHER, S. K., MCEWEN, K. M., HALVERSON, D. A., JACOBS, D. T., and LINDNER, J. (1993). Self-organized criticality : an experiment with sandpiles. *Am. J. Phys.*, 61(4):329–335.

HEUDIN, J.-C. (1994). *La vie artificielle.* Hermès.

HILLIS, W. D. (1988). *La Machine à connexions.* Masson.

HUANG, J., NARKOUNSKAIA, G., and TURCOTTE, D. L. (1992). A cellular-automata, slider-block model for eathquakes II. Demonstration of self-organized criticality for a 2-D system. *Geophys. JJ. Int.*, pages 259–269.

JAEGER, H. M. and NAGEL, S. R. (1992). Physics of the granular state. *Science*, 255:1481–1612.

JEZEQUEL, F. (1996). *Parallélisation en temps et en espace de la résolution d'équations d'évolution, contrôle de la solution.* Thèse de doctorat, Université Pierre et Marie CURIE, Paris VI.

JOLIVET, L. (1995). *La déformation des continents : exemples régionaux.* Enseignement des sciences. Hermann.

JUTEAU, T. (1993). *La naissance des océans.* Editions Payot & Rivages.

KEAREY, P. and VINE, F. J. (1996). *Global Tectonics.* Blackwell Science Ltd.

KINCAID, C. and OLSON, P. (1987). An experimental study of subduction and slab migration. *J. of Geophysical Research*, 92(B13):13,832–13,840.

LAFAURIE, B. (1995). *Modélisation de la convection par une méthode de gaz sur réseau et technique de suivi d'interface.* Thèse de doctorat, Université Paris XI Orsay. N° d'ordre 3622.

LALEVÉE, P. (1995). *Algorithmes parallèles par flux dans les graphes : des fondements aux applications.* Thèse de doctorat, Université Pierre et Marie CURIE, Paris VI.

LALLEMAND, S. (1992). *Transferts de matière en zone de subduction : réflexion sur les conséquences de l'érosion tectonique (vol 2).* Mémoire d'habilitation à diriger des recherches, Université Pierre et Marie CURIE, Paris VI.

LALLEMAND, S. and MALAVIEILLE, J. (1992). L'érosion profonde des continents. *La Recherche*, 23(249):1388–1397.

LALLEMAND, S. E., SCHNÜRLE, P., and MALAVIEILLE, J. (1994). Coulomb theory applied to accretionary and nonaccretionary wedges : possible causes for tectonic erosion and/or frontal accretion. *J. of Geophysical Research*, 99(B6):12,033–12,055.

LANGLOIS, A. and PHILIPPS, M. (1997). *Automates cellulaires : application à la simulation urbaine.* Collection Sytèmes Complexes. Hermès.

LAUDON, J. and LENOSKI, D. (1999). The SGI Origin : a ccNUMA highly scalable server. http://www.sgi.com/origin/numa.html.

LEDUC, T. (1997). Modélisation par un système dynamique discret du processus de subduction-érosion en tectonique des plaques : première approche uni-dimensionnelle. Rapport interne, Laboratoire LIP6, ftp://ftp.lip6.fr/lip6/reports/1997/lip6.1997.008.ps.gz.

LEDUC, T. (1998a). A one-dimensional discrete computer model of the subduction erosion phenomenon. In *CESA'98 Nabeul-Hammamet.* (2e Congrés Mondial IMACS et IEEE/SMC).

LEDUC, T. (1998b). Parallélisation d'Automates Cellulaires uni- et bi-dimensionnels et application à la modélisation du processus de subduction-érosion en tectonique des plaques. In *RenPar'10 Strasbourg.*

LEDUC, T. (1999). Simulation par automates cellulaires bi-dimensionnels en géotectonique. *Technique et Science Informatique*, 18(4):397–420.

LE GUYADEC, Y. (1995). *Etude sémantique du modèle de programmation data-parallèle ; application à la preuve de programmes et à la conception de schémas de compilation.* Thèse de doctorat, Université d'Orléans.

LE PICHON, X. (1968). Sea floor spreading and continental drift. *J. of Geophysical Research*, 73(12).

LE PICHON, X., FRANCHETEAU, J., and BONNIN, J. (1973). *Plate tectonics.* Elsevier Scientific Pub. Co.

MALAVIEILLE, J., LARROQUE, C., and CALASSOU, S. (1993). Modélisation expérimentale des relations tectonique/sédimentation entre bassin avant-arc et prisme d'accrétion. *C.R. Acad. Sci. Paris*, 316, Série II:1131–1137.

MARGOLUS, N. (1984). Physics-like models of computation. *Physica D*, (10):81–95.

MIGNOT, J.-C., DESPREZ, F., TOURANCHEAU, B., ROBERT, Y., REYMANN, O., PRYLLI, L., DION, M., and DOMAS, S. (1996). Programmation parallèle sur machines à mémoire distribuée et réseaux de stations de travail. Formation LIP—ENS LYON.

MÜLLER, D. (1996). *Techniques informatiques efficaces pour la simulation de milieux granulaires par des méthodes d'éléments distincts.* Thèse de doctorat, EPFL. Dépt de Mathématiques - Ecole Polytechnique Fédérale de Lausanne - thèse numéro 1545.

NARKOUNSKAIA, G. and TURCOTTE, D. L. (1992). A cellular-automata, slider-block model for eathquakes I. Demonstration of chaotic behaviour for a low order system. *Geophys. JJ. Int.*, pages 250–258.

PACHECO, P. S. (1995). A user's guide to MPI. ftp://math.usfca.edu/pub/MPI. Department of Mathematics - University of San Francisco.

RAFFIN, B. (1997). *Un modèle structuré de communication et de synchronisation pour le parallélisme de tâches.* Thèse de doctorat, Université d'Orléans.

ROBERT, F. (1995). *Les systèmes dynamiques discrets.* Mathématiques et Applications 19. Springer-Verlag.

RÓKA, Z. (1994). *Automates Cellulaires sur graphes de Cayley.* Thèse de doctorat, ENS Lyon - Université Claude Bernard - Lyon I.

SCHATTEN, A. (1997). Cellular Automata (Digital Worlds). http://qspr03.tuwien.ac.at/~aschatt/info/ca/ca.html.

SCHNÜRLE, P. (1994). *Contribution à la compréhension des mécanismes dérosion tectonique et à la quantification des flux de matière dans les zones de subduction.* Thèse de doctorat, Université Pierre et Marie CURIE, Paris VI.

SCHOLL, D. W., von HUENE, R., VALLIER, T. L., and HOWELL, D. G. (1980). Sedimentary masses and concepts about tectonic processes at underthrust ocean margins. *Geology*, 8:564–568.

SHEMENDA, A. I. (1994). *Subduction : Insights from physical modelling.* Kluwer Academic Publishers.

SHEMENDA, A. I. and GROCHOLSKY, A. L. (1992). Physical modelling of lithosphere subduction in collision zones. *Tectonophysics*, 216:273–290.

SILICON GRAPHICS (1999a). *Origin 2000 and Onyx 2 Performance Tuning and Optimization guide.* Document technique numéro 007-3430-002. http://techpubs.sgi.com/library.

SILICON GRAPHICS (1999b). *Origin 2000 rack - guide de l'utilisateur (hardware).* Document technique numéro 007-3456-003FR. http://techpubs.sgi.com/library.

SILICON GRAPHICS (1999c). *Origin 2000 rackmount owner's guide.* Document technique numéro 007-3456-003. http://techpubs.sgi.com/library.

SMITH, R. (1991). The application of cellular automata to the erosion of landforms. *Earth Surface Processes and Landforms*, 16:273–281.

SNIR, M., OTTO, S., HUSS-LEDERMAN, S., WALKER, D., and DONGARRA, J. (1995). *MPI: The Complete Reference.* Scientific and Engineering Computation series. MIT Press. http://www.netlib.org/utk/papers/mpi-book/mpi-book.html.

STERN, H. (1995). *Pratique de NFS et NIS.* Editions O'Reilly International Thomson. Traduction de Managing NFS and NIS par Céline VALOT.

TANG, C. and BAK, P. (1988). Critical exponents ans scaling relations for self-organized critical phenomena. *Phys. Rev. Lett.*, 60(23):2347–2350.

VICHNIAC, G. Y. (1984). Simulating physics with cellular automata. *Physica D*, (10):96–116.

von HUENE, R. and CULLOTTA, R. (1989). Tectonic erosion at the front of the Japan convergent margin. *Tectonophysics*, 160:75–90.

von HUENE, R. and LALLEMAND, S. (1990). Tectonic erosion along the Japan and Peru convergent margins. *Geological Society of America Bulletin*, 102:704–720.

von HUENE, R. and SCHOLL, D. W. (1991). Observations at convergent margins concerning sediment subduction, subduction erosion, and the growth of continental crust. *Rev. Geophys.*, 29(3):279–316.

WOLFRAM, S. (1986). Cellular Automaton fluids 1 : basic theory. *J. Stat. Phys.*, 45(3/4):471–526.

WOLFRAM, S. (1994). *Cellular Automata and Complexity.* Addison-Wesley Publishing Company.

Index