Les algorithmes de la modélisation : une analyse critique pour la modélisation économique

Buda, Rodolphe

GAMA-MODEM, Université de Paris 10

July 2001

Université de Paris X - Nanterre

LES ALGORITHMES DE LA MODÉLISATION

UNE PRÉSENTATION CRITIQUE POUR LA MODÉLISATION ÉCONOMIQUE

Rodolphe BUDA
GAMA-MODEM, Université de Paris X-Nanterre

DOCUMENT DE RECHERCHE - MODEM N°2001-44 Juillet 2001

"*Cette prétention de pouvoir accroître la puissance de l'esprit humain par un contrôle conscient de sa croissance se fonde ainsi sur la thèse qui déclare également pouvoir pleinement expliquer cette croissance ; elle implique la possession d'une sorte de super-esprit de la part de ceux qui la soutiennent* [...]". **Friedrich August Von HAYEK**, *Scientisme et sciences sociales*, 1953, (trad. Paris, Plon, p.142).

"[...] *The market may be considered as a computer sui generis which serves to solve a system of simultaneous equations. It operates like an analogue machine ; a servo-mechanism based on the feedback principle. The market may be considered as one of the oldest historical devices for solving simultaneaous equations* [...]". **Oskar Ryszard LANGE**, "The Computer and the Market", in C.H.FEINSTEIN ED., *Socialism, Capitalism and Economic Growth*, Cambridge UP, 1967, p.159.

LES ALGORITHMES DE LA MODÉLISATION

UNE PRÉSENTATION CRITIQUE POUR LA MODÉLISATION ÉCONOMIQUE[1]

Rodolphe BUDA
GAMA-MODEM[2]-Université de Paris X-Nanterre

RÉSUMÉ : *A travers la présentation du large et très riche éventail d'algorithmes qui sont (ou pourraient être) utilisés en modélisation économ(étr)ique, notre papier tente de mettre en évidence les sources d'erreurs d'interprétation voire d'erreurs conceptuelles que pourraient entraîner un usage trop "aveugle" de l'algorithmique. Ce n'est pas tant les faiblesses structurelles de l'outil (précision déficiente de l'arithmérique des ordinateurs) que le risque que prendrait le modélisateur en oubliant que toutes ses représentations, quelles qu'elles soient (centralisées ou non etc.), sont et seront toujours des artefacts dans le champ de la réalité économique et sociale. Il ne faut certainement pas proscrire le recours à l'algorithmique pour autant, dès lors que l'on prévoit des procédures qui ramènent le système de calculs à la réalité (par expérimentation).*

MOTS-CLÉS : *Computational Economics, Modélisation macroéconométrique, Simulation, Algorithmes.*

SUMMARY : *Our aim is to specify all the kind of errors and mistakes which come from an unreasonable use of algorithmic. So we examine a large and rich set of algorithms, some are used in economic modelling others would be. We can observe some structural error (accuracy of computer), but the worse error comes from the belief we would all represent with algorithms. To make a good use of algorithms in social sciences, we have to introduce procedures which can lead us to the reality, not only statistics but experiment too.*

KEY-WORDS : *Computational Economics, Macroeconomic Modelling, Simulation, Algorithms.*

[1]- Ce papier a été rédigé dans le cadre du Chap.2, "Modélisation et problèmes algorithmiques" de notre thèse de Doctorat. Nous ferons ainsi référence à des modules (ESTIME, GEBANK, PROGEN+COMBIN etc.) que nous avons programmés dans le cadre de la construction d'un logiciel de modélisation (SIMUL) - voir Annexe I.

[2]- **G**roupe d'**A**nalyse **M**acroéconomique **A**ppliquée, 200, Avenue de la République, 92001 NANTERRE Cedex - FRANCE - Tél. : 01-40-97-77-88 - E-mail : *rodolphe.buda@u-paris10.fr*
MODEM : **M**odélisation de la **D**ynamique **E**conomique et **M**onétaire.

Introduction Générale

L'objet de ce papier n'est pas tant de présenter les principaux algorithmes utilisés en modélisation économique - nombre de manuels font des présentations de meilleure qualité et plus exhaustives - que d'en proposer une vision critique. Les modèles économiques, et plus particulièrement les modèles macroéconométriques, sont des représentations numériques qui, de ce fait, ont opéré des choix de simplification voire de réduction de la réalité. Revenir sur les algorithmes existants peut donc, nous l'espérons, constituer une étape vers la reformulation d'algorithmiques plus féconds pour la modélisation[3] - voir Fig.1.

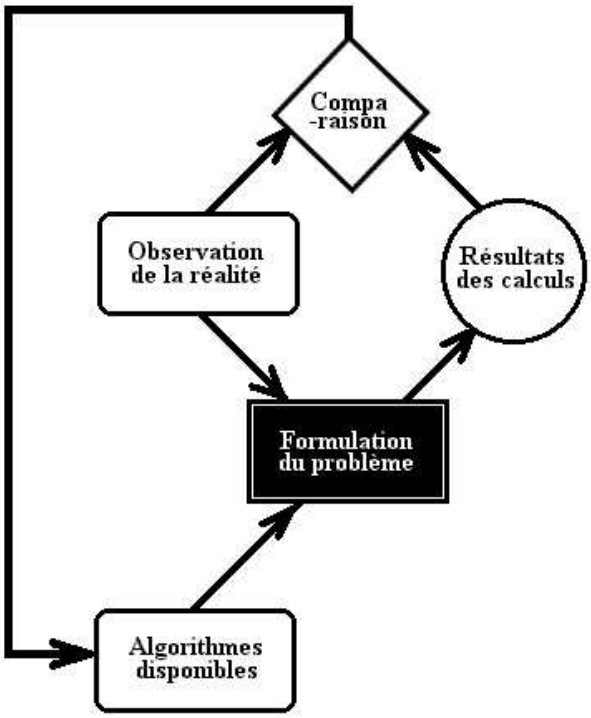


Fig.1 - Formulation des modèles et réexamen des algorithmes[4]

Le problème de la modélisation consiste à se poser la question de savoir, compte tenu de l'état observé de l'économie et sous certaines hypothèses, quelle sera en mode projection, quelle serait (en mode simulation), l'état futur (vs l'état alternatif) de cette économie ? Depuis la phase de gestion de la banque de données qui requiert divers algorithmes de tri, jusqu'aux algorithmes d'analyse nu-

[3]- Voir les désillusions suscitées par les modèles macroéconométriques in P.ARTUS et al. (1986, pp.242-44). Ce réexamen s'impose particulièrement, selon nous, en ce qui concerne les équations de comportement des modèles macroéconométrique - voir notre note (1997.b).

[4]- La modélisation peut être considérée comme une démarche d'observation de la réalité (analyse) puis une formulation du problème à résoudre (synthèse). Les résultats obtenus grâce au modèle peuvent alors être comparés à la réalité. L'une des conséquences peut alors être le perfectionnement des algorithmes.

mérique impliqués dans les calculs matriciels d'estimation économétrique - pour être bref -, le fonctionnement de la modélisation macroéconométrique s'explique par des algorithmes[5]. Il implique l'emploi d'une syntaxe, l'algorithmique, et d'un langage, les mathématiques. L'algorithme est une séquence d'instructions ordonnées et formalisées, permettant d'aboutir à la résolution du problème étudié. Peu d'ouvrages sont consacrés aux phases algorithmiques de la modélisation[6].

Si les algorithmes visent tous à assister la décision (analyses rétrospective et prospective), ils sont loin de former une librairie homogène de programmes. Nous aborderons des algorithmes directement liés à un traitement numérique (estimation statistique, simulation optimisation). Mais nous consacrerons également quelques lignes à des algorithmes de nature apparemment "moins numériques", mais intervenant dans des phases déterminantes de la modélisation. Il s'agira d'une part des algorithmes permettant de structurer et/ou d'analyser des données ainsi que des algorithmes graphiques et ceux de communication. Enfin nous aborderons brièvement le problème de précision des calculs lié à l'arithmétique des ordinateurs. Délibérément, nous n'avons développé les aspects relatifs au Génie logiciel[7], de même que dans un souci de clarté, nous avons regroupé les programmes en annexe, lorsque la compréhension n'exigeait pas qu'ils accompagnent le texte. Notre présentation sera jalonnée de travaux algorithmiques et de références à nos notes de travail, réalisés dans le cadre de notre thèse de Doctorat - voir en Annexe I, la présentation générale du projet.

RÉFÉRENCES

ALMON C., (1967), *Matrix Methods in Economics*, Reading (Mass.), Addison-Wesley, 164 p.

ARTUS P., DELEAU M., MALGRANGE P., (1986), *Modélisation macroéconométrique*, Paris, Economica, Coll.Economie et statistiques avancées, 283 p.

BRILLET J.L., (1994), *Modélisation économétrique - principes et techniques*, Paris, Economica, Coll.Economie et statistiques avancées, 194 p. + **Le logiciel Soritec Sampler.**

DELEAU M., MALGRANGE P., (1978), *L'analyse des modèles macroéconométriques quantitatifs*, Paris, Economica, Coll.Economie et statistiques avancées, 256 p.

FAIR R.C., (1996), "Computational Methods for Macroeconometrics Models", in H.M.AMMAN & D.A.KENDRICK, *Handbook of Computational Economics*, Amsterdam, North-Holland, pp.143-70.

[5]- Les algorithmes que nous allons présenter ne sont pas propres à l'économie mathématique. La Physique recourt à des algorithmes de simulation depuis longtemps. La Météorologie ne peut procéder que de cette manière. Enfin, citons les Biomathématiques qui traduisent les mécanismes métaboliques sous formes de modèles de simulation ou d'optimisation, grâce à des équations différentielles, notamment - voir à ce propos Y.CHERRUAULT (*Biomathématiques*, Paris, PUF, Coll. Que sais-je ?, 1983, pp.19-20, ainsi que pp.86-106).

[6]- Signalons toutefois C.ALMON (1967) qui fournit à la fois algorithmes et programmes, M.DELEAU & P.MALGRANGE (1978) qui proposent une analyse des modèles macroéconométriques français ainsi que la méthodologie de la modélisation. Sur la programmation de procédures algébriques appliquées à la macroéconomie et à la comptabilité nationale on pourra consulter J.F.PHÉLIZON (1979). La méthodologie est décrite par P.ARTUS et al. (1986) qui y apportent une touche théorique, alors que P.JACQUINOT et al. (1991) et J.L.BRILLET (1994) fournissent une présentation à l'adresse des praticiens. Citons enfin R.C.FAIR (1996) qui présentent les méthodes de simulation et d'optimisation utilisée en macroéconométrie - ses programmes en FORTRAN sont disponibles sur internet.

[7]- A propos des principes du génie logiciel que nous avons examiné dans le cadre de notre travail de programmation du logiciel SIMUL, voir nos notes de travail (1993.a ; 1993.b ; 1993.c ; 1994.a et 1995.b). Le lecteur intéressé par le génie logiciel est invité à consulter I.SOMMERVILLE (*Le génie logiciel*, Paris, Addison-Wesley, 1992, 638 p.) pour un exposé développé et complet et/ou J.PRINTZ (*Le génie logiciel*, Paris, PUF, Coll. Que sais-je ?, 128 p., 1995) pour un exposé plus concis.

JACQUINOT P., LOUFIR A., MIHOUBI F., (1991), *Muscadet et Muscadine - deux outils pour la micro-informatique appliquée à la macro-économie*, Paris, Economica, 230 p. + **Les logiciels Muscadet et Muscadine.**

PHÉLIZON J.F., (1979), *Traitement statistique des données*, Paris, Economica, 242 p. + **Programmes.**

NOTES DE TRAVAIL

(1992), "Nécessité d'un module cartographique en modélisation régionale", *Mimeo GAMA*, Université de Paris X-Nanterre, oct. + **Le module GEOGRA.**

(1993.a), "Optimisation et rationalisation des algorithmes du système intégrés de régression multi- dimensionnelles", *Mimeo GAMA*, Université de Paris X-Nanterre, fév.

(1993.b), "Analyse des possibilités d'implémentation d'algorithmes de modélisation macro- économiques", *Mimeo GAMA*, Université de Paris X-Nanterre, juill.

(1993.c), "Réflexions sur l'architecture d'un logiciel de modélisation macroéconométrique", *Mimeo GAMA*, Université de Paris X-Nanterre, sept.

(1994.a), "Note complémentaire relative au choix du langage de programmation d'un système de modélisation macro-économique multi-dimensionnelle", *Mimeo GAMA*, Université de Paris X-Nanterre, juil. + **Le programme LARGEMAT.**

(1994.b), "Réflexions au sujet de l'algorithme de transformation des séries des modules PROGEN & COMBIN", *Mimeo GAMA*, Université de Paris X-Nanterre, oct.

(1994.c), "Essai de modélisation des communications entre agents économiques", *Mimeo GAMA*, Université de Paris X-Nanterre, nov. + **Le logiciel MEREDIT.**

(1994.d), "Modules de transformation des séries PROGEN et COMBIN - passage en séries multi- dimensionnelles", *Mimeo GAMA*, Université de Paris X-Nanterre, déc. + **Le module PROGEN.**

(1995.a), "GEBANK 1.0 - Module de gestion des banques de données multi-dimensionnelles", *Mimeo GAMA*, Université de Paris X-Nanterre, juil. + **Le module GEBANK.**

(1995.b), "Problèmes algorithmiques en modélisation multi-dimensionnelle", *Mimeo GAMA*, Université de Paris X-Nanterre, juil. + **La procédure MANTISSE.**

(1996.a), "Proposition for Aggregation's Algorithms in Multi-Regional Economic Modeling", *Mimeo GAMA*, Université Paris X-Nanterre, jan. + **Le programme AGREG.**

(1996.b), "Construction d'un logiciel de modélisation multi-dimensionnel - présentation générale d'un système, SIMUL 2.1 et examen des principaux problèmes", *Mimeo GAMA*, Université de Paris X-Nanterre, juin + **Le module CHRONO.**

(1996.c), "Présentation d'un outil de contrôle de la précision des calculs en modélisation macro- économétrique", *Mimeo GAMA*, Université de Paris X-Nanterre, août + **Le logiciel GNOMBR.**

(1997.a), "De la pertinence de la précision astronomique dans la mesure du temps en dynamique économique", *Mimeo GAMA*, Université de Paris X-Nanterre, avr.

(1997.b), "La modélisation macroéconomique comme processus de communication - réflexions pour une formalisation finaliste des équations de comportement", *Mimeo GAMA*, Université de Paris X-Nanterre, mai. (première version juin 1994).

(1998), "De la précision arithmétique des ordinateurs - proposition d'algorithmes de calculs scientifiques de haute précision", *Mimeo GAMA*, Université de Paris X-Nanterre.

(1999.a), SIMUL - Manuel de références et guide d'utilisation version 3.1, *Mimeo GAMA*, Université de Paris X-Nanterre, 60 p. + **Le système SIMUL.**

(1999.b), "Market Exchange Modelling - Experiment, Simulation Algorithms, and Theoretical Analysis", *Communication in Experimental Economics - ESA, Grenoble*, 7-8 oct. + **Les logiciels SINGUL et ECHANGE.**

(2000.a), "Un bref historique de l'économie expérimentale", *Mimeo GAMA*, Université de Paris X- Nanterre, Séminaire du Modem du 20 avril.

(2000.b), "Modélisation économique quantitative vs individualisme méthodologique ?", *Communication au Colloque de l'AHTEA : Quelles perspectives pour une économie autrichienne appliquée ?*, Paris, 18-19 mai.

SOMMAIRE

0 - INTRODUCTION 1

RÉFÉRENCES 2
NOTES DE TRAVAIL 3
SOMMAIRE 4
ANNEXE I - LE PROJET DE SYSTÈME DE MODÉLISATION SIMUL 6

I - LES ALGORITHMES DE SIMULATION ET D'OPTIMISATION 7

a) GÉNÉRALITES SUR L'ANALYSE NUMÉRIQUE 7
b) LES ALGORITHMES DE LA SIMULATION 9
i - Les méthodes "géométriques" de résolution de systèmes d'équations 10
La méthode dichotomique 10
La méthode de la sécante (dite de Lagrange ou des parties proportionnelles) 10
La méthode de la tangente (dite de Newton) 11
ii - Les méthodes "algébriques" de résolution de systèmes d'équations 11
La méthode Gauss-Seidel 12
Les méthodes de relaxation, d'élimination et de triangularisation 13
iii - Les algorithmes de simulation basés sur des lois de probabilité 14
Génération de nombres aléatoires pour la simulation 14
Dérivée et intégrale numérique 15
c) LES ALGORITHMES DE L'OPTIMISATION 17
i - La programmation linéaire 18
Énoncé du problème 18
L'algorithme du Simplexe (dite de Dantzig) 19
La programmation en nombres entiers et la programmation mixte 20
ii - La programmation non linéaire 21
Les algorithmes de l'optimisation sans contrainte 21
Méthode du gradient (dite de plus grande pente, dite de Cauchy) 21
Méthode de Newton-Raphson 21
Les algorithmes de l'optimisation sous contraintes 22
La méthode du multiplicateur de Lagrange 22
Les méthodes de programmation convexe de Beale, Dantzig et Rosen 23
Le contrle optimal 24
iii - Les principales applications de la recherche opérationnelle 24
La notion d'arbre et de graphe 24
Problème de transport 25
Ordonnancement 25
Gestion de stocks et des files d'attente 26
Programmation dynamique 27
Jeux 27
RÉFÉRENCES 28
ANNEXE 1.1 - PROGRAMMES D'ANALYSE NUMÉRIQUE ET DE SIMULATION 30
ANNEXE 1.2 - PRÉSENTATION DU MODULE RESOLV 33

II - LES ALGORITHMES D'ORGANISATION, DE GESTION ET D'ANALYSE DES DONNÉES 35

a) LES ALGORITHMES D'ORGANISATION DES DONNÉES 35
i - Les algorithmes de tri et de recherche 35
Algorithmes de tri 35
Recherche de données 36
ii - Les algorithmes de transformation de données 37
La transformation simple des données 37
L'agrégation des données 38
iii - Les algorithmes de reconstitution de données 38
Données manquantes et interpolation 39
Données manquantes, équilibrage de tableaux avec marges connues 40
iv - Le calcul matriciel sur grands tableaux 41
La technique des "matrices creuses" 42
La technique du produit par blocs 43
La technique des "matrices-disque" 44

b) LES ALGORITHMES D'ANALYSE STATISTIQUE DE DONNÉES 45
i - L'analyse de données 45
L'analyse en composantes principales 45
Les méthodes dérivées de l'A.C.P. 47
ii - L'économétrie 48
Moindres carrés ordinaires 48
Les techniques dérivées 49
RÉFÉRENCES 52
ANNEXE 2.1 - PROGRAMMES DE GESTION DE DONNÉES 53
ANNEXE 2.2 - ÉTUDE DE L'ALGORITHME
D'AGRÉGATION VECTORIELLE 55

**III - LES ALGORITHMES GRAPHIQUES ET
LES ALGORITHMES DE COMMUNICATION** 60

a) LES ALGORITHMES GRAPHIQUES 60
i - Les procédures graphiques usuelles 60
Algorithmes de discrétisation 62
Algorithme du peintre 63
ii - Les algorithmes de maillage, de pavage et de construction d'espaces 63
Algorithmes de pavage 64
Algorithmes fractals 64
Logique floue 65
b) LES ALGORITHMES DE COMMUNICATION 65
i - Réseaux locaux, protocoles et communication entre ordinateurs 65
Principe du Token ring 66
Algorithme de Dekker-Peterson 67
Algorithme du sémaphore 67
ii - Les applications économiques sur réseaux locaux et sur internet 68
Expérimentation et échanges d'informations 68
Agent-Based Computational Economics 69
Vers des algorithmes de marché 70
RÉFÉRENCES 71
ANNEXE 3.1 - LES MÉCANISMES DE MARCHÉ DU MODÈLE SINGUL 72
ANNEXE 3.2 - LES ALGORITHMES DE MARCHÉ DU MODÈLE SINGUL 73

IV - PRÉCISION DES CALCULS ET ARITHMÉTIQUE DES ORDINATEURS 74

a) LES PRINCIPALES APPLICATIONS DES ALGORITHMES
DE L'ARITHMÉTIQUE 74
i - Généralités sur le codage 74
Changement de base 74
Clé de sûreté 75
Les algorithmes de compression des données 75
Algorithme de Shannon-Fano 75
Algorithme de Huffman 76
Algorithme de Ziv-Lempel 76
ii - La cryptologie 76
La cryptographie RSA clé publique 76
iii - Précision calendaire 81
b) LE CONTROLE DE LA PRÉCISION DES CALCULS SUR ORDINATEURS 84
i - L'énoncé du problème 84
La norme IEEE-754 dite arithmétique en virgule flottante 84
ii - Les pistes de contrle algébrique 85
Le conditionnement 85
Algorithme de Horner 85
iii - Les pistes de contrôle arithmétique 86
Les arithmétiques stochastiques 86
L'arithmétique d'intervalles 86
Les arithmétiques en multiprécision et les arithmétiques exactes 86
Algorithmes d'artithmétique en multiprécision 86
Algorithmes d'arithmétique exacte 89
Les arithmétiques dynamiques 89
Algorithme d'Avizienis 89
iv - Problème ergonomique de lisibilité des mantisses de résultats 89
RÉFÉRENCES 90
ANNEXE 4.1 - CALCUL D'ERREUR D'ARRONDI AVEC
LE LOGICIEL GNOMBR 91
ANNEXE 4.2 - APPLICATION DU LOGICIEL GNOMBR
A UN TES SIMPLIFIÉ 92
ANNEXE 4.3 - PROCÉDURE DE REMANTISSAGE INTELLIGIBLE
DES RÉSULTATS DE CALCULS 93
ANNEXE 4.4 - PRÉSENTATION DES ALGORITHMES SCOLAIRES IMPLÉMENTÉS DANS GNOMBR

V - CONCLUSION 94
RÉFÉRENCES 96
ANNEXE 5 - LES ÉCONOMISTES ET LES MACHINES A CALCULER 97

ANNEXE I - LE PROJET DE SYSTÈME DE MODÉLISATION SIMUL[8]

Le système SIMUL se compose de trois grandes catégories de modules. D'une part, les modules de gestion des données et résultats (préparation, calcul, etc.) : GEBANK (gestion des banques de données), GRAPHE (gestion des graphiques), PROGEN-COMBIN (calculs et transformations de données) et SIMBNK (simulation de banques de données).

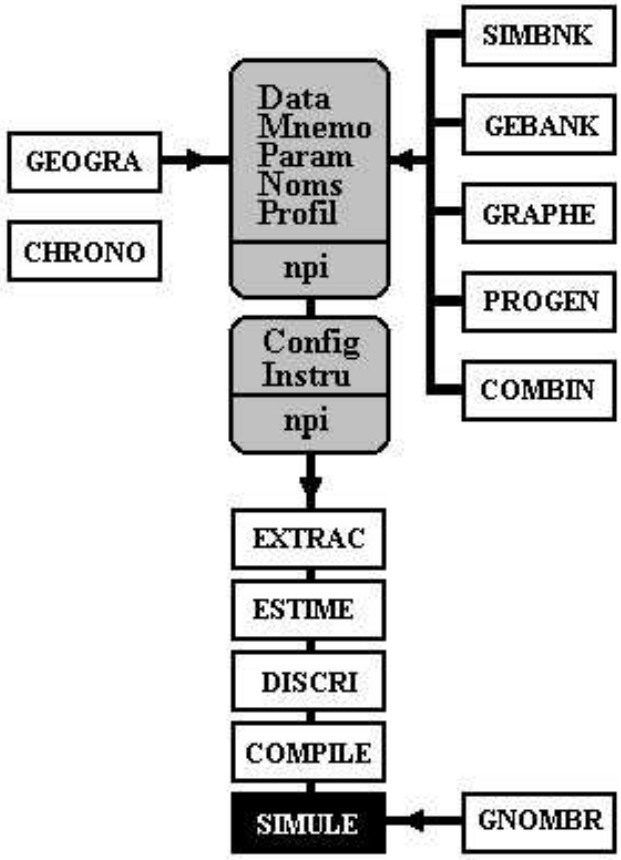


Fig.I.a - Vue d'ensemble des modules de SIMUL[9]

D'autre part, les modules d'analyse économétrique : EXTRAC (constitution des séries à estimer), ESTIME (estimation économétrique) et DISCRI (tri des équations selon des critères statistiques fournis par le modélisateur). Enfin les modules périphériques, qui permettent d'accomplir des opérations en dehors d'une session de modélisation - *i.e.* collecte de données, construction du modèle et simulations - : GEOGRA (cartographie des données régionales), GNOMBR

[8]- D'après notre *Manuel de références SIMUL 3.1.*

[9]- BANK, BANQUE, COMPILE, EQUAT, REGILINK, REGINA, REGIS, SIMULE, TRI © Raymond Courbis, Gérard Cornilleau, Jari Mansinen & GAMA, mcmlxxv-mmiii.
AGREG, CHRONO, COMBIN, DISCRI, ECHANGE, EXTRAC, ESTIME, LARGEMAT, GEBANK, GEOGRA, GNOMBR, GRAPHE, GRECAL, MANTISSE, MEREDIT, PROGEN, RESOLV, SIMBNK, SIMUL, SINGUL © Rodolphe Buda & GAMA, mcmxcii-mmiii.

(permet d'analyser la précision des calculs) et CHRONO (permet une chronologie plus fine des séries par le calcul des jours ouvrables français). Les modules CHRONO et GNOMBR fonctionnent indépendamment du système de données (DATA, MNEMO, PARAM, NOMS, PROFIL et NPI suffixe correspondant au nom du pays), tandis que SIMBNK fonctionne en amont sans le système de données (il génère en effet un système de données fictives). Les autres modules chercheront en revanche tout ou partie des fichiers de données pour fonctionner. Signalons enfin que le système SIMUL a été conçu au départ pour fonctionner avec des modèles de la classe REGIS (REGIS, REGINA et REGILINK[10]) - les modules COMPILE et SIMULE appartiennent au système initial mais sont appelés à être remplacés par le module RESOLV en cours de construction.

[10]- Voir à ce propos R.COURBIS (ED.), *Modèles régionaux et modèles régionaux-nationaux*, Paris, Cujas, Coll. GAMA, 1979, 370 p.

I/ Les Algorithmes de Simulation et d'Optimisation

Le rôle d'un algorithme est de permettre d'aboutir à une solution à partir des paramètres du problème. Quel que soit l'objectif général sous-tendu par le problème, à savoir la simulation - *i.e.* la reproduction artificielle et numérique d'un phénomène donné - ou l'optimisation - *i.e.* la recherche de la meilleure solution parmi toutes celles possibles -, les mathématiques nous fournissent des outils communs dans une branche spécialisée : l'analyse numérique. Les phénomènes qui intéressent le modélisateur sont rarement représentables de manière aussi simple ; le nombre de dimensions du problème dépasse en général l'unité[1]. La simulation consiste à reproduire un phénomène donné, uniquement selon les caractéristiques qui intéressent le modélisateur. Les techniques de simulation et d'optimisation sur ordinateur peuvent avoir des applications très vastes[2]. Après avoir rappelé le rôle de l'analyse numérique, nous décomposerons l'ensemble des algorithmes disponibles en deux groupes, ceux relatifs à la simulation et enfin ceux employés en recherche opérationnelle - *i.e.* l'optimisation.

a) GÉNÉRALITÉS SUR L'ANALYSE NUMÉRIQUE

Les résultats classiques de l'analyse et de l'algèbre ne sont pas toujours exploitables numériquement, directement sur ordinateur. Soit que les caractéristiques des ordinateurs l'interdisent - rappelons que l'analyse de données n'a pu être programmée que près de deux cents ans après avoir été découverte - (problème de capacité mémoire et de temps d'accès requis, etc.), soit que le cheminement ne soit pas optimal - un ordonnancement différent d'opérations mathématiques permettant d'obtenir le même résultat plus rapidement. L'application informatique directe des calculs proposés par l'analyse et l'algèbre n'est pas possible, sans accepter un degré d'imprécision si faible soit-il. Le rôle de l'analyse numérique consiste donc à permettre ce passage de la "théorie mathématique du calcul" à son application. L'analyse numérique est une branche des mathématiques, qui se présente comme un ensemble de techniques permettant d'aboutir à la résolution numérique de problèmes quantitatifs. Elle est apparue non pas avec l'ordinateur, mais plutôt avec de nouveaux besoins en calculs. La Comptablité[3]- comme technique de mesure de la richesse -, l'Architecture[4] - comme technique de mesure des proportions des édifices - ou l'Astronomie[5] -

[1]- Les premiers modèles programmés sur ordinateurs présentaient entre dix et douze inconnues. Progressivement ce nombre est monté au delà de la centaine pour atteindre le pic de la dizaine de milliers, selon le degré de désagrégation des économies représentées.

[2]- A propos d'un panorama assez détaillé et documenté sur la question, voir notamment B.JOLIVALT (*La simulation et ses techniques*, Paris, PUF, Que-sais je ?,1995). L'auteur présente essentiellement des simulations de pilotage professionnel sur ordinateur, ainsi que des simulations destinées aux loisirs. Bien que les simulations économiques n'y figurent pas, on y trouve des références intéressantes. Voir en particulier la simulation en réseau (pp.95-99).

[3]- Si l'on en croit les historiens de la Comptabilité, elle serait née en Mésopotamie - pour plus d'informations voir J.G. DEGOS, *Histoire de la comptabilité*, Paris, PUF, Coll. Que sais-je, 1998, pp.7-20.

[4]- A propos du calcul des proportions dans les arts de l'espace, voir M. CLEYET-MICHAUD, *Le nombre d'or*, Paris, PUF, Coll. Que sais-je ?, pp.98-122, 1973, (Rééd.1993) - notamment à propos de la polémique autour du nombre d'or qui aurait été utilisé par les Egyptiens lors de la construction des pyramides.

[5]- A propos des calculs de trajectoires des astres, voir notamment J.MEEUS, 1986, *Calculs astronomiques à l'usage des amateurs*, Paris, Société Astronomique de France, 152 p. A propos

comme technique de mesure du temps - ont "consommé" des calculs très tôt dans l'Histoire de l'Humanité. Cependant, leurs calculs n'impliquaient pas des algorithmes complexes et se limitaient souvent à l'usage d'opérations arithmétiques élémentaires et à des fonctions trigonométriques de base. Certes les civilisations antiques ont contribué à l'édification de l'Algèbre moderne ; les Babyloniens au XVIIIè siècle avant notre Ere, étaient parvenus à résoudre des systèmes de plusieurs équations à plusieurs inconnues du premier et du second degré par des méthodes géométriques (G.IFRAH, 1981, Tome 2, pp.453-56), au Ier siècle de notre Ere, les Chinois par des méthodes proches du calcul matriciel actuel (G.IFRAH, 1981, Tome 1, pp.662-65), puis au IV-Vè siècle de notre Ere, les Indiens avaient enrichi la représentation du zéro et des nombres négatifs. Mais les méthodes proposées n'étaient alors pas généralisables.

TABLEAU N°1 - Contributions à la syntaxe et à la sémantique de l'analyse numérique[6]

INNOVATIONS OU DÉCOUVERTES	ANNÉES	AUTEURS	PAYS
signes + et -	1489	J.W.d'EGER	ALLEMAGNE
Racine carrée	1525	C.RUDOLFF	ALLEMAGNE
Nombres imaginaires	1545-1560	G.CARDANO et R.BOMBELLI	ITALIE
symbole =	1557	R.RECORDE	ANGLETERRE
Notation prédécimale	1582	S.STEVIN	BELGIQUE
Notation algébrique	1591	F.VIETE	FRANCE
Notation prédécimale	1592	J.B RGI	SUISSE
Notation décimale	1592	G.A.MAGINI	ITALIE
symboles < et >	1631	T.HARIOT	ANGLETERRE
signe *	1632	W.OUGHTRED	ANGLETERRE
Notation exponentielle	1637	R.DESCARTES	FRANCE
Notation exponentielle	1656	J.WALLIS	ANGLETERRE
Symbole infini mathématique	1656	J.WALLIS	ANGLETERRE
Calcul différentiel	1672	G.W.LEIBNITZ	ALLEMAGNE
Etude générale des fonctions	1748	L.EULER	SUISSE
Etude système d'équations	1750	G.CRAMER	SUISSE
Théorie des substitutions	1770	A.VANDERMONDE	FRANCE
Intégrales elliptiques	1786	F.LEGENDRE	FRANCE
Théorie des fonctions analytiques	1797	L.LAGRANGE	FRANCE
Limite et borne d'un nombre	1799	K.F.GAUSS	ALLEMAGNE
Séries trigonométriques	1807-1822	J.FOURIER	FRANCE
Notion de continuité	1821	A.CAUCHY	FRANCE
Solution des équations différentielles	1821	A.CAUCHY	FRANCE
Convergence des séries	1830	A.CAUCHY	FRANCE
Algèbre de Boole	1854	G.BOOLE	ANGLETERRE
Calcul matriciel	1858	A.CAYLEY	ANGLETERRE
Nombres irrationels et réels	1872	R.DEDEKIND	ALLEMAGNE

Le calcul numérique n'a pu progresser qu'à partir du moment où d'une part, les mathématiciens avaient amélioré la représentation des nombres qu'ils proposaient (apparition du zéro, des nombres négatifs, de la représentation décimale, etc.) et d'autre part, à partir du moment où ceux-ci avaient généralisé les résultats grâce à une formalisation littéraire abstraite - voir TABLEAU N°1. L'analyse numérique peut se concevoir finalement comme un "art du calcul"

du calculs du temps, voir P. COUDERC, (*Le calendrier*, Paris, PUF, Coll. Que sais-je ?, 1946, (Rééd.1993).

[6]- D'après la chronologie proposée par G.IFRAH (*op.cit.*, Tome 2, 1981, pp.458-67). Pour une présentation historique très documentée, accompagnée de commentaires sur les articles originaux, voir également J.L.CHABERT et al. (EDS), *Histoire d'algorithmes*, Paris, Belin, Coll.Regards sur la science, 591 p.

(voir R.THÉODOR, 1989). Elle propose en effet des techniques de calculs qui arbitrent constamment entre deux critères, à savoir la rapidité et l'approximation des calculs. Ces grandes têtes de chapitres "généralistes"[7] sont la recherche de solutions à des problèmes d'analyse, la recherche de solutions optimales, le calcul matriciel, les techniques de convergence vers la solution et les techniques d'approximation[8]. Il s'agit à chaque fois d'obtenir un résultat numérique (et non pas une formulation algébrique) le plus précis possible et le plus rapidement possible. On peut distinguer plusieurs chapitres dans cette discipline : 1 - La recherche de solutions à des problèmes d'analyse consiste à effectuer la résolution de systèmes d'équations linéaires et non linéaires, la résolution d'équations différentielles, ainsi que l'intégration et la dérivation numériques - *i.e.* par opposition à ces mêmes opérations dans le sens fonctionnel du terme[9]. 2 - La recherche de solutions optimales consiste à utiliser des méthodes telles que celle du gradient ou du simplexe, etc. qui déterminent la solution minimale ou maximale d'un système donné. 3 - Le calcul matriciel est un ensemble de transformations qui donnent les caractéristiques d'une matrice (calcul de déterminant, valeurs propres, valeurs singulières, vecteurs propres, etc.) ou qui affecte la valeur de ses éléments (factorisation, méthode de triangularisation, conditionnement de matrices[10], etc.). 4 - Les techniques de convergence vers la solution consistent à rechercher et affiner l'intervalle où se situe la solution (recherche et accélération de la convergence, recherche par dichotomie, tests d'arrêts des itérations, discrétisation, transformation, relaxation, etc.). 5 - Les techniques d'approximation consistent d'une part, à transformer des fonctions de formes peu pratiques à résoudre en une combinaison additive et/ou multiplicative de fonctions élémentaires (méthodes d'approximation, méthodes d'interpolations, lissage, etc.), d'autre part à déterminer des algorithmes accédant plus rapidement aux données significatives (matrices creuses, propagation d'erreurs d'arrondi, etc.)[11]. Cela étant, les progrès de l'analyse numérique sont loins d'être achevés - voir le TABLEAU N°1 non exhaustif, qui recense les principales contributions de l'analyse numérique. D'une part en raison des récents travaux concernant la théorie des Objets fractals, la théorie du Chaos ou la théorie des sous-ensembles flous[12] qui introduisent de nouveaux types de problèmes à résoudre ; d'autre part

[7]- La plupart des manuels présentent ces grandes têtes de chapitres. Certains ouvrages spécialisés traitent de points précis de l'analyse numérique tels que la représentation des nombres et des fonctions par les ordinateurs et l'incidence sur la précision des calculs ou bien encore certains algorithmes spécifiques (Cf. Infra).

[8]- Citons les trois ouvrages : P.LASCAUX et R.THÉODOR (1986-87, 2 Vol.) qui proposent des démonstrations et des algorithmes en langage mathématique des principaux résultats ; M. LA PORTE et J.VIGNES (1974-80, 2 Vol.) propose également les démonstrations des résultats ainsi que des organigrammes informatiques et des programmes en FORTRAN 77 ; M.SIBONY (1988) et M.SIBONY & J.C.MARDON (1982-88, 2 Vol.) proposent des démonstrations et des organigrammes des principaux résultats.

[9]- En l'occurence la recherche de la solution d'une intégrale dont les bornes sont finies et non pas la recherche de primitives d'une fonction.

[10]- Ce terme comprend plusieurs acceptions, selon les cas il s'agit d'une technique permettant de rendre une matrice inversible, diagonalisable, ou un système soluble.

[11]- Il est impossible de fournir une liste exhaustive des programmes gratuits ou non, d'analyse numérique. En dehors des ouvrages que nous citons, et pour lesquels nous avons fait figurer la mention "+ Programmes" ou "+ Logiciel" dans les références bibliographiques, nous invitons le lecteur à visiter les sites internet des centres de recherches (INRIA, MIT, etc.). D.DUREISSEIX (*Méthodes numériques appliquées à la conception par éléments finis*, Mimeo, ENS Cachan, 2000, 74 p.) propose quant à lui de se référer à E.ANDERSON et al. (1999).

[12]- A partir des années cinquante, l'extension de l'analyse économique statique aux dimen-

en raison de l'apparition des ordinateurs neuronaux qui exigeront de nouveaux algorithmes de calculs en parallèle.

b) LES ALGORITHMES DE LA SIMULATION

Le problème général de base de la simulation consiste à calculer, si elle existe, la valeur de x telle que l'on ait $f(x) = 0$ - où f est une fonction et x l'inconnue et, par extension des systèmes d'équations à plusieurs inconnues. On distingue les méthodes "géométriques" des méthodes "algébriques" - basée sur le calcul matriciel. Cependant, d'autres phénomènes nécessitent des représentations basées sur le recours à des lois de probabilité.

i - Les méthodes "géométriques" de résolution de systèmes d'équations

Elles sont basées sur la recherche "géométrique"[13] des coordonnées de points se rapprochant de celle de la solution de $f(x) = 0$ - où f est linéaire ou non.

La méthode dichotomique

Supposons que la solution de $f(x) = 0$ se trouve dans l'intervalle de recherche $[a, b]$ - voir Fig.2 -, alors il est facile d'en déduire que $f(A)$ et $f(B)$ n'auront pas le même signe. En d'autres termes, on aura $\Pi = f(A).f(B) < 0$.

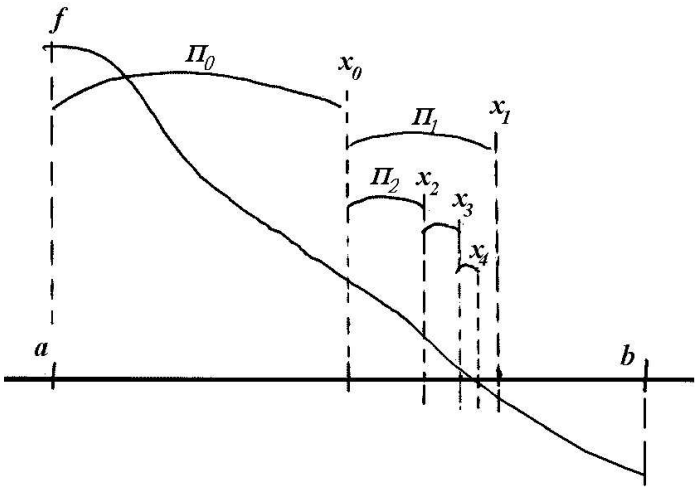


Fig.2 - Recherche dichotomique de $f(x) = 0$

sions temporelle d'une part puis spatiale d'autre part, a été l'occasion d'un bouleversement méthodologique important de la discipline - voir C.PONSARD (ED.) (*Analyse économique spatiale*, Paris, PUF, Coll.Economie, 1988, pp.193-230) à propos des sous-ensembles flous appliqués à l'économie spatiale, E.PUMAIN (ED.) (*Analyse spatiale et dynamique des populations*, Paris, J.Libbey-Ined, 1991, 457 p.) pour l'analyse chaotique spatiale, G.ABRAHAM-FROIS (ED.) ("La dynamique chaotique", *Revue d'économie politique*, 1994, N 2/3) ainsi que G.ABRAHAM-FROIS et E.BERREBI (*Instabilité, cycles, chaos*, Paris, Economica, 1995, 392 p.) pour les analyses chaotique et fractales temporelles essentiellement.

[13]- Il s'agit là d'un abus de langage dans la mesure où les méthodes algébriques peuvent également avoir une interprétation géométrique, mais avec plus de trois dimensions.

On divise ensuite l'intervalle en deux $[a, x0[$ et $[x_0, b]$. On peut chercher dans lequel des deux intervalles se manifeste le changement de signe en calculant Π_0. On subdivise alors le nouvel intervalle $[x_0, b]$ en deux $[x_0, x_1[$ et $[x_1, b]$ et on calcule Π_1. Le changement de signe se manifeste dans $[x_0, x_1]$, ce qui permet alors de déterminer la nouvelle partition de recherche $[x_0, x_2[$ et $[x_2, x_1]$ etc. L'algorithme s'arrête lorsque la taille de l'intervalle est inférieure à la valeur d'un ε fixée à l'avance - voir le programme en Turbo-Pascal en Annexe 1.1.

La méthode de la sécante (dite de Lagrange ou des parties proportionnelles[14])

Si l'intervalle de recherche est $[a, b]$ - voir Fig.3 -, on commence par déterminer les coordonnées du point d'intersection $(x_0, f(x_0))$ entre l'axe des abscisses et la droite $(a, f(a))(b, f(b))$. Ces coordonnées s'obtiennent comme

$$x_0 = \frac{b.f(a) - a.f(b)}{f(a) - f(b)}$$

et $f(x_0)$. On choisit alors l'intervalle qui maintient le changement de signe - dans notre exemple l'intervalle $(x_0, f(x_0))(b, f(b))$. On poursuit la recherche jusqu'à l'itération k, au point de coordonnées[15]

$$x_k = \frac{b.f(x_{k-1}) - x_{k-1}.f(b)}{f(x_{k-1}) - f(b)}$$

et $f(x_k)$, tel que la différence entre x_{k-1} et x_k soit inférieure à la valeur d'un fixée à l'avance - voir le programme en Turbo-Pascal en Annexe 1.1.

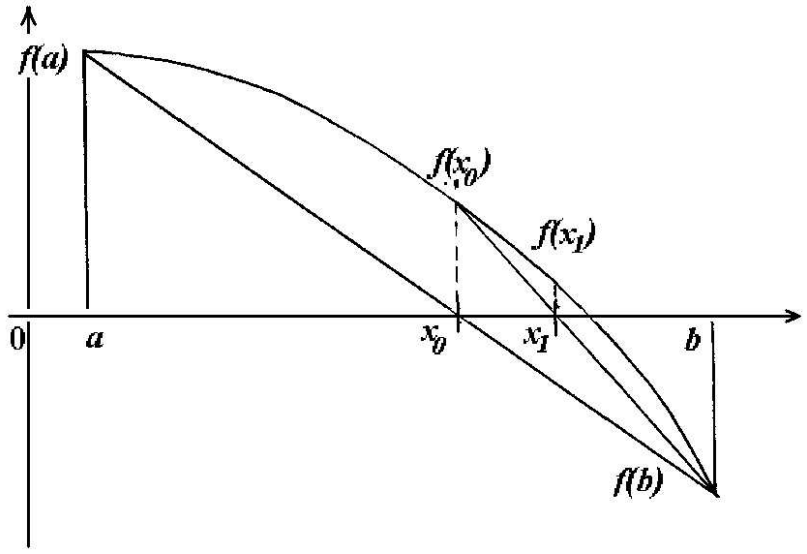


Fig.3 - Recherche de $f(x) = 0$ par la sécante

[14]- Voir D.MONASSE (1988) à propos de son lien avec le théorème des accroissements finis.
[15]- Dans notre exemple la sécante est au dessous de la courbe. Dans le cas contraire on a :

$$x_k = \frac{a.f(x_{k-1}) - x_{k-1}.f(a)}{f(x_{k-1}) - f(a)}$$

La méthode de la tangente (dite de Newton)

Cette méthode consiste à choisir un point de départ $(x_0, f(x_0))$ puis à projeter la droite de tangente en ce point sur l'axe des abscisses. On obtient alors un nouveau point de coordonnées $(x_1, f(x_1))$.

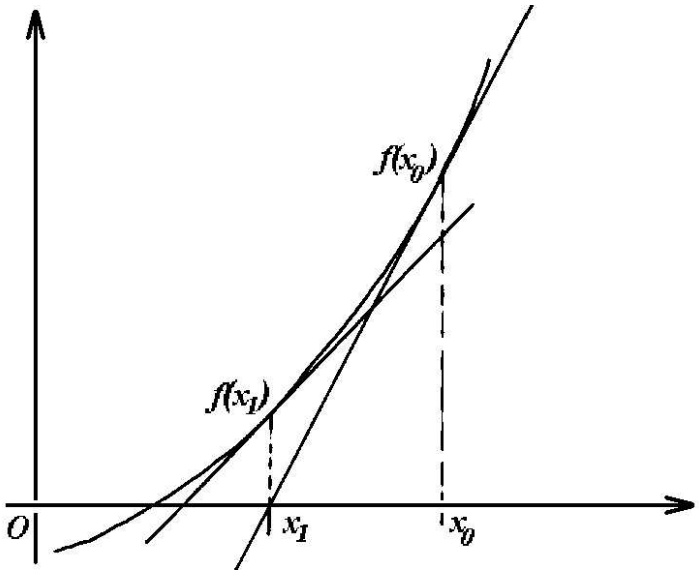


Fig.4 - Recherche de $f(x) = 0$ par la méthode de la tangente[16]

A l'ordre k, la tangente

$$y = f'(x_k).x + f(x_k) - x_{0k}.f'(x_{0k})$$

coupe l'axe des abscisses en

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

La programmation de cette méthode est simple et ce programme figure dans toutes les bibliothèques d'analyse numérique - voir le programme en Turbo-Pascal en Annexe 1.1.

ii - Les méthodes "algébriques" de résolution de systèmes d'équations

Ces méthodes[17] consistent quant à elles, à recombiner par substitutions successives des éléments de la matrice du système d'équations linéaires pour parvenir à la simplification donnant la valeur de chacune des inconnues. Plusieurs algorithmes de résolution fonctionnent selon ce principe. Nous développerons

[17]- Voir panorama des méthodes et preuves J.G.DION et R.GAUDET (1996, pp.304-413). Consulter J.BERSTEL et al. (1991, tome 1) à propos des méthodes et des programmes en Pascal, ainsi que A.REVERCHON et M.DUCAMP (1994, pp.113-269) pour des programmes en C++.

uniquement la méthode Gauss-Seidel, de loin la plus utilisée par les modélisateurs en raison de sa simplicité[18] puis nous évoquerons les autres méthodes. Bien que présentée dans tous les manuels d'algèbre, la méthode du Déterminant : soit

$$A.x = b$$

on a

$$A^{-1} = \frac{1}{\det(A)}(A^*)^t$$

et

$$a^*_{ij} = (-1)^{signe\ \sigma}.\det(A_{ij})$$

où A^* est la matrice des cofacteurs. Cette méthode n'est jamais appliquée car trop gourmande en calculs : le temps de calculs est de 11 s pour la taille 15, de 433 jours pour la taille 20, de 11 millions d'années pour la taille 25 (J.G.DION et R.GAUDET, *op.cit.*, pp.313-18).

La méthode Gauss-Seidel

Soit A la matrice des coefficients du système d'équations, b le vecteur des constantes et x le vecteur des inconnues, le système s'écrit : $A.x = b$. La méthode Gauss-Seidel - J.K.F.GAUSS (1826) et L.SEIDEL (1874) - consiste à démarrer d'une approximation initiale de la solution, pour estimer la solution du système par corrections successives du résidu $r^{(k)} = b - A.x^{(k)}$, où k est l'ordre de résolution - *i.e.* la k-ème itération. A partir d'un vecteur initial $x^{(0)}$, on va calculer les vecteurs successifs $x_i^{(k+1)}$ comme suit :

$$x_i^{(k+1)} = \frac{1}{a_{ii}}.\left(b_i - \sum_{j=1}^{i-1} a_{ij}.x_j^{k+1} - \sum_{j=i+1}^{N} a_{ij}.x_j^k\right)$$

On parle de convergence du système lorsque la différence entre une même inconnue X à l'ordre k et $k+1$ est inférieure à un seuil de convergence fixé à l'avance. La convergence est dite "globale" si toute la valeur de toutes les composantes atteignent leur seuil en même temps, "locale" si une seule composante l'atteint en premier, et "semi-globale" si quelques composantes l'atteignent en premier. On calcule la différence notée δx_i^{k+1} de la manière suivante :

$$\delta x_i^{(k+1)} = x_i^{(k+1)} - x_i^{(k)}$$

$$\delta x_i^{(k+1)} = \frac{1}{a_{ii}}.\left(b_i - \sum_{j=1}^{i-1} a_{ij}.x_j^{k+1} - \sum_{j=i}^{N} a_{ij}.x_j^k\right)$$

Cependant, la convergence n'est pas toujours assurée. Si l'on suppose que l'écriture du système précédent revient à poser que l'on cherche les solutions de $X = f(X)$, quatre cas sont possibles - voir tableau ci-après et Fig.5.

[18]- Parce qu'elle économise de la place mémoire et qu'elle converge plus rapidement que les autres méthodes. Gauss-Seidel nécessite un tableau (celui de la matrice A), alors que la méthode de Jacobi en nécessite deux - P.LASCAUX et R.THÉODOR (1987, *op.cit.*, pp.406-09).

CAS	FORMULE	COMPORTEMENT
A	$0 < \frac{\delta f(X)}{\delta X} < 1$	CONVERGENCE MONOTONE
B	$\frac{\delta f(X)}{\delta X} < -1$	DIVERGENCE CYCLIQUE
C	$-1 < \frac{\delta f(X)}{\delta X} < 0$	CONVERGENCE CYCLIQUE
D	$\frac{\delta f(X)}{\delta X} > 1$	DIVERGENCE MONOTONE

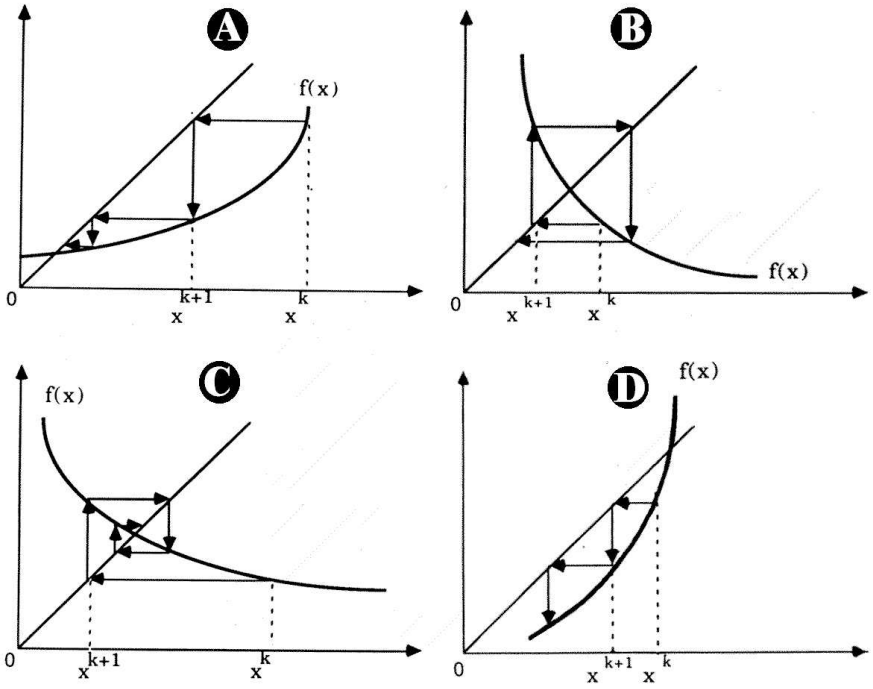


Fig.5 - La méthode de résolution Gauss-Seidel[19]

La séquence de programmation s'écrit comme suit :

```
POUR I :=1 N FAIRE
  S :=0
  POUR J :=1 N FAIRE
    S :=S+A[I,J]*X[J]
  FIN DE BOUCLE J
  R :=B[I]-S
  X[J] :=X[J]+R/A[I,I]
FIN DE BOUCLE I
```

Fig.6 - Programmation d'une itération de l'algorithme Gauss-Seidel

Les méthodes de relaxation, d'élimination et de triangularisation

Les méthodes de Gauss-Seidel et de Jacobi (1846) dont le terme général s'écrit comme suit :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{i-1} a_{ij}.x_j^k \right)$$

sont des algorithmes appartenant à une forme générale d'algorithmes dont le terme général s'écrit :

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{i-1} a_{ij}.x_j^k \right) + (1-\omega).x_i^k$$

où ω est un terme de relaxation (R.SOUTHWELL, 1946) qui doit être choisi pour permettre une plus rapide convergence de la résolution[20].

```
POUR I=1 A N FAIRE
  S=B[I]
  POUR J=1 A I-1 FAIRE
    S=S-A[I,J]*X[j]
  FIN J
  POUR J=I+1 A N FAIRE
    S=S-A[I,J]*X[J]
  FIN J
  Y[I]=S/A[I,I]
FIN I
POUR I=1 A N FAIRE
  X[I]=Y[I]
FIN I
```

Fig.7 - Programmation d'une itération de l'algorithme Jacobi

Citons enfin les méthodes d'élimination telle que la méthode de Gauss-Jordan ou celle d'Elimination de Gauss[21]. Ces algorithmes consistent à choisir un pivot k, de la matrice A - *i.e.* la ligne k et la colonne k simultanément - à partir duquel on modifie les coefficients de la matrice jusqu'à obtenir une matrice triangulaire[22]. Certaines factorisation permettent d'économiser de la place mémoire - par exemple la factorisation LU ("lower-upper"). Lorsque la matrice A est symétrique définie positive, alors on peut lui appliquer un algorithme plus rapide : celui d'A.L.CHOLESKY (Cdt BENOIT, 1924). On trouvera en Annexe 1.1 les procédures que nous avons implémentées dans la bibliothèque de notre

[20]- A noter que le choix de l'amorçage des itérations n'est pas sans conséquences sur la rapidité de convergence, notamment en raison du Théorème de Brower (dit "*du point fixe*") selon lequel, si une application f est continue alors $\exists\ x\ /\ f(x) = x$.

[21]- K.F.GAUSS (1826, *op.cit.*) cité par G.B.DANTZIG (1966, p.44). Voir chez ce dernier (Chap.2) l'exposé relatif aux méthodes d'élimination appliquées aux systèmes d'équations et aux systèmes d'inéquations.

[22]- Voir à ce propos P.LASCAUX et R.THÉODOR (1986, tome 1, pp.207-87).

module provisoire de résolution de systèmes d'équations linéaires, RESOLV[23] - voir Annexe 1.2.

iii - Les algorithmes de simulation basés sur des lois de probabilité

La simulation de certains phénomènes économiques ou de gestion nécessite l'implémentation de fonctions de base telles que les tests statistiques usuels et la génération de lois statistiques (uniforme, normale, exponentielle, binomiale, hypergéométrique, Gamma, Bernoulli, Pascal, Poisson[24]). De plus les outils de différenciation (dérivées, intégrales) s'avèrent également être nécessaires à l'implémentation de procédures de simulation.

Génération de nombres aléatoires pour la simulation

Une méthode couramment utilisée est la Méthode de Monte-Carlo[25]. On souhaite simuler des événements (arrivée de clients à des caisses etc.) dont la probabilité P est connue. On effectue un tirage "aléatoire" d'une variable r (par ex. : les décimales de Π , les chiffres d'une horloge d'ordinateur etc.). Si $r \leq P$ alors on décide que l'événement a lieu, autrement on décide que l'événement n'a pas lieu[26]. Toutefois, le "calcul" de nombres aléatoires[27] n'est pas sans soulever d'importantes questions de fond. Il y a en effet une totale antinomie entre la qualification d'aléatoire et la détermination fonctionnelle de ces nombres, de sorte qu'il paraît impossible de générer des nombres purement aléatoires[28]. Ainsi, par exemple, 1° - la sélection de décimales du nombre Π

[23]- Le module n'est pas intégré au système pour le moment. Cependant, il permet de résoudre les systèmes d'équations avec trois méthodes différentes Gauss-Seidel, Élimination de Gauss et Gauss-Jordan.

[24]- Voir à ce propos J.F.PHÉLIZON (1977) et C.V.FEUVRIER (1971, pp.60-88).

[25]- Voir à ce propos J.M. HAMMERSLEY et D.C. HANDSCOMB (Monte Carlo Methods, London, Chapman Hall, 1964). Pour des développements opérationnels et des programmes voir F.Y.BOIS et D.R.MASLE (1997).

[26]- On peut déterminer la probabilité d'événement à partir des deux équations suivantes :

$$H = \alpha_0 + \sum_{i=1}^{N} \alpha_i . X_i$$

et

$$P = \frac{1}{1 - e^{-H}}$$

où la relation entre H et les N grandeurs est obtenue par estimation économétrique. Les grandeurs X_i sont choisies en raison de leur forte connexion au phénomène dont on évalue la probabilité de survenue. La "probabilité" P est obtenue par le calcul d'une fonction logistique sur H. Voir à ce propos le chapitre 16 - The Maths of Microsimulation in CORSIM (2000, pp.169-83).

[27]- Une méthode classique consistait à calculer une suite de nombres h_i tels que

$$h_i = k.h_{i-1} + c \mod t$$

en choisissant h_0, k et c judicieusement, on allonge au maximum la période de cette suite périodique. Voir à ce propos R.FAURE, 1979, pp.186-88), J.F.PHÉLIZON (*op.cit.*, pp.62-64) pour des programmes FORTRAN ainsi que M.ABRAMOVITZ et I.A.STEGUN (*Handbook of Mathematical Functions*, U.S. Dept. of Commerce, 1966) pour un panorama plus détaillé. Pour des développements mathématiques et informatiques, voir D.E.KNUTH (1981, tome 2, pp.1-177). Voir en Annexe 1.1 notre fonction de tirage d'événement aléatoire dont on connaît la probabilité de survenue, en langage en turbo-Pascal.

[28]- Voir H.LEEB (1995, pp.70-95) à propos de l'étude des "imperfections" des principaux

n'est pas sans risque, car rien ne dit que tous les chiffres soient représentés de manière équiprobable dans la partie fractionnaire de Π ; 2° - La qualité de la sélection des chiffres d'une horloge d'ordinateur dépend de sa fréquence, or celle-ci étant automatisée, elle biaise nécessairement le tirage ; 3 - Le tirage mécanique (du type tirage des boules de loto) trahit, sur le long terme, les nécessaires imperfections physiques des éléments mécaniques[29].

Dérivée et intégrale numérique

Les outils mathématiques issus de l'analyse mathématique, tels que la dérivation, l'intégration et le calcul différentiel, permettent de représenter des phénomènes tels que le stockage, les files d'attente ou bien encore des phénomènes de durée de vie et de mortalité propres aux modèles démographiques. Sauf à faire du calcul formel, il n'est pas question de déterminer la forme fonctionnelle d'une dérivée ou d'une primitive pour l'appliquer à une valeur donnée - d'autant que les fonctions ne sont pas toujours intégrables. Les algorithmes de dérivation se déduisent sans aucun problème de leur forme analytique, aussi bien pour la dérivée première que pour la dérivée seconde d'une fonction f - voir en Annexe 1.1. En ce qui concerne l'intégrale

$$I = \int_a^b f(x).dx$$

d'une fonction entre deux bornes a et b, la formulation intuitive qui vient à l'esprit consiste à découper l'intervalle $[a, b]$ en N assez grand, puis à calculer une somme de N rectangles. Le calcul de l'intégrale devient donc où

$$S = \Delta x. \sum_{i=0}^{N-1} f(x_i)$$

est la somme des N rectangles de taille $\Delta x.f(x_i)$. L'algorithme de Simpson améliore cette méthode et obtient une convergence plus rapide[30]. Alors que ce dernier type d'algorithme fournit la valeur de l'intégrale bornée, il existe des algorithmes permettant de résoudre les équations différentielles - *i.e.* trouver la fonction $y = f(x)$ telle que

$$\sum_{n=1}^{N} \frac{d^n y}{dx} = 0$$

La méthode la plus fréquemment utilisée, celle de Runge-Kutta (C.RUNGE, 1895 ; W.KUTTA, 1901), est basée sur des interpolations linéaires successives - déterminée par un développement en série de Taylor - autour d'une solution particulière[31]. On notera que les manuels ne proposent aucun algorithme particulier

générateurs de nombres aléatoires.

[29]- Les sociétés de jeux de hasard changent régulièrement leur matériel.

[30]- On pourrait en effet démontrer qu'une méthode plus rapide consiste à chercher non plus des rectangles, mais des trapèzes. Pour des développements plus importants sur les méthodes d'intégration numérique (Newton-Cotes et Gauss-Legendre), voir J.G.DION et R.GAUDET (*op.cit.*, pp.261-84) ainsi que M.SIBONY et J.C.MARDON (tome 2, *op.cit.*, Chap.IV).

[31]- A propos des aspects théoriques, voir M.SIBONY et J.C.MARDON (tome 2, *op.cit.*, Chap.V) ainsi que R.THÉODOR (*op.cit.*, pp.227-88). Pour un aperçu et des programmes en Pascal, voir D.MONASSE (*op.cit.*, pp.188-205), en C++ voir A.REVERCHON et

pour résoudre les équations de récurrence de la forme

$$\sum_{n-1}^{N} a_n.x_n = 0$$

Tous ces "algorithmes différentiels" sont indirectement utiles pour la simulation, puisqu'ils permettent d'évaluer les "fonctions structurantes" du système au voisinage de l'état recherché. Ainsi, la dérivation peut être nécessaire dans le cadre d'une résolution systématique de système d'équations par la méthode de Newton. En ce qui concerne le calcul des intégrales bornées, celui-ci est requis pour simuler les processus de vie des individus au sein d'une population, en faisant l'hypothèse que le temps est une variable continue[32]. La même logique intervient pour la simulation des files d'attentes et requiert la résolution d'équations différentielles[33]. Les problèmes, en général plutôt théoriques[34], du type calcul des paramètres de "politique de stabilisation" en économie fermée, font également intervenir la résolution d'équations différentielles[35]. Les modèles systémiques de J.W. FORRESTER, dans la mesure où ils représentent l'économie comme des systèmes plus ou moins auto-régulés, recourt également aux équations différentielles[36]. Enfin, les modèles dynamiques empiriques à temps discret sont souvent plus pertinents que ceux à temps continu, eu égard à la forme des statistiques disponibles[37]. Les comptes nationaux sont en effet, au mieux mensuels, mais on ne peut pas parler des données en temps continu[38].

M.DUCAMP (*op.cit.*, pp.271-401), en Basic voir R.DONY (1986). Pour un exposé et une comparaison des autres méthodes (Euler, Adams-Bashforth, Adams-Moulton) voir J.G.DION et R.GAUDET (*op.cit.*, pp.537-54).

[32]- Ainsi par exemple, si l'on suppose une fonction p telle que $p(x,t)$ est la probablité de survie à la naissance à l'âge x et à la date t, $\phi(x,t)$ la fécondité féminine instantanée à la date t - pour les naissances féminines seulement -, alors on a

$$N(t) = \int_{\alpha}^{\beta} N(t-x).p(x).\phi(x).dx$$

$\forall t > \beta$ où N est la fonction de "reproduction féminine" durant la période génésique $[\alpha, \beta]$ - voir à ce propos J.AMEGANDJIN, *Démographie mathématique*, Paris, Economica, Coll. Economie et statistiques avancées, 1989, pp.207-15.

[33]- En effet, le problème consiste à déterminer la probabilité P de survenue de l'événement : arrivée d'un élément x dans la file d'attente durant l'intervalle $t + \Delta t$, sachant que les probabilités d'entrée et de sortie (resp.) de la file d'attente sont $\lambda\Delta t$ et $\mu\Delta t$ (resp.). Si l'on pose

$$P(X_{t+\Delta t} = x) = f(x)$$

alors lorsque $\Delta t \to 0$, on obtient

$$\frac{df(x)}{dt} = \lambda.f(x-1) - (\lambda+\mu).f(x) + \mu.f(x+1)$$

Voir à ce propos J.F.PHÉLIZON (*op.cit.*, pp.201-205).

[34]- Voir G.ABRAHAM-FROIS et E.BERREBI (*op.cit.*, 1995).

[35]- Pour un exposé de la résolution des équations différentielles dans le cadre de l'analyse des politiques économique, voir M.C.BARTHÉMY (*Mathématiques des systèmes dynamiques*, Paris, Dalloz, Coll. Mémento, 1989, pp.109-32). A propos du modèle, voir A.W.PHILLIPS, "Stabilisation Policy in a Closed Economy", *Economic Journal*, june, 1954.

[36]- Voir J.W.FORRESTER (1968, *Principes des systèmes*, Lyon, PUF, Coll. Sciences des systèmes, pp.6-13, trad. 1984). Le langage Dynamo des années soixante soixante-dix a été remplacé par un langage "visuel" dans le programme VenSim PLE Version 4.

[37]- Sur les temps discret et continu, voir G.GANDOLFO (*Economic Dynamics : Methods and Models*, Amsterdam, North-Holland, 1980, 571 p.).

[38]- Cela étant, il est néanmoins toujours possible d'effectuer des interpolations afin de pro-

Pour conclure provisoirement et de manière non exhaustive sur la question des algorithmes de simulations, ajoutons que certains algorithmes obéissent à des règles ad hoc ; citons ainsi le fameux "Jeu de la vie" de J.H.CONWAY[39].

c) LES ALGORITHMES DE L'OPTIMISATION

Alors que les algorithmes de la simulation économique répondaient à la question "Quel devrait être, sous certaines hypothèses, l'état d'une réalité économique systématisée ?, les algorithmes de l'optimisation raisonnent en sens inverse. La question posée est en effet, "A quelles conditions le système atteint-il un état déterminé ? Pour parvenir à formuler le problème, la technique générale de l'optimisation consiste à représenter par des fonctions - *i.e.* des relations d'égalités ou d'inégalités linéaires ou non linéaires - le système économique dans l'espace des variables économiques pertinentes - *i.e.* un plan, un espace ou un hyperespace selon le nombre de variables. La démarche suivie par les algorithmes d'optimisation consistent alors à rechercher en mode maximisation ou minimisation (resp.) les coordonnées du point situé dans la partie supérieure ou inférieure (resp.) du domaine formé par les différentes fonctions. La fonction que l'on cherche à rendre maximale (ou minimale) est appelée la "fonction objectif", tandis que les autres fonctions, lorsqu'elles existent dans le problème, s'appellent les "contraintes" ; les contraintes les plus évidentes étant les contraintes de positivité[40]. L'ensemble de ces techniques et algorithmes est rassemblé dans une discipline appelée "Recherche opérationnelle"[41]. Nous l'avons dit, la programmation linéaire occupe une part importante de la littérature de la Recherche opérationnelle, car elle constitue une technique de base. Bien que la plupart du temps les problèmes non linéaires puissent être transformés en problèmes linéaires, des algorithmes plus complexes ont néanmoins dû être mis au point pour résoudre des problèmes non linéaires : ces algorithmes sont rassemblés dans la programmation non linéaire. Nous verrons enfin quelles sont les principales applications de la recherche opérationnelle (gestion de stocks, minimisation de flots, ordonnancement etc.)[42]. Il est couramment admis que la Recherche opérationnelle fasse partie des disciplines de gestion, cependant ces techniques sont néanmoins appliquées en économie. Dans ces cas là, il s'agit alors de proposer des alternatives aux simulations de politiques économiques[43] et de proposer une

poser des modèles en temps continu, mais les hypothèses supplémentaires ne sont pas anodines sur l'analyse des résultats.

[39]- J.H.CONWAY (*Scientific American*, Oct. 1970, p.120) : On répartit de manière aléatoire des individus sur une surface constituée de cellules. On instaure des lois dynamiques de reproduction et de mortalité des individus (ex. : si un individu est isolé il meure, si deux individus sont séparés par un espace, un troisième individu apparaît). Selon les conditions initiales, la population peut s'accroître ou au contraire s'éteindre très rapidement. Les recherches de J.H.CONWAY ont succédé celles de J. Von NEUMANN parues en 1978 ("The General and Logical Theory od Automate", in BUCKLEY (ED.), *Modern System Research for the Behavioural Scientist*, Chicago, Adline Pub.), après la mort de ce dernier.

[40]- Les quantités de marchandises par exemple ne peuvent être négatives.

[41]- Voir la définition fournie par R.FAURE (1979, *op.cit.*, pp.1-10), ainsi que V.COHEN (*Recherche opérationnelle*, Paris, PUF, Coll. Que sais-je ?, 1995, 128 p.) pour un panorama plus large.

[42]- La plupart des bibliothèques de programmes proposent des algorithmes de calculs d'optimisation, mais il existe des logiciels spécifiques tels que GINO et LINDO (L.SCHRAGE, 1989), et des langages spécifiques, tels que DATAFORM de MANAGEMENT SCIENCE SYSTEMS (1970) ou bien encore AMPL (R.FOURER et al., 1990).

[43]- Voir à ce propos M.GUILLAUME (*Modèles économiques*, Paris, PUF, Coll.Thémis,

réponse à la question Quelle politique doit être menée pour permettre d'obtenir un état déterminé de l'économie nationale[44] ? De plus dans une perspective de planification où l'information des prix n'est pas disponible, nous verrons que les techniques de l'optimisation peuvent s'avérer très utiles[45].

i - La programmation linéaire

Le cas le plus simple à résoudre, et en même temps le cas de base, est naturellement celui où les fonctions à représenter sont linéaires (fonction objectif et contraintes).

> "*Le résultat de l'élaboration du modèle est ainsi l'ensemble des relations mathématiques caractérisant tous les programmes possibles pour le système. Cet ensemble constitue le modèle de programmation linéaire. Une fois que le modèle est établi, le problème de programmation linéaire peut être posé sous sa forme mathématique, et sa solution peut être interprétée comme un programme d'action pour le système, c'est-à-dire comme une suite d'actions permettant au système réel d'évoluer de son état initial vers l'objectif requis.*
> *Le problème de programmation linéaire revient donc à déterminer des niveaux pour toutes les activités du système de telle sorte qu'ils :*
> *a) ne soient pas négatifs,*
> *b) satisfassent aux équations d'équilibre, et*
> *c) conduisent à l'efficacité la plus grande possible.*"
> (**G.B.DANTZIG**, 1966, p.4).

Énoncé du problème

Considérons par exemple une entreprise qui fabrique deux biens en quantités x_1 et x_2. Les contraintes techniques de fabrication des produits sont exprimées en heures d'usinage dans deux ateliers - a_{ji} est la durée d'usinage du produit i dans l'atelier j et a_{jmax} est la durée maximale d'utilisation de l'atelier j.

$$\sum_{i=1}^{N} a_i^j . x_i \leq 0$$

De plus les quantités x_1 et x_2 ne peuvent être négatives $x_i \leq 0$. On détermine ainsi la zone de production possible hachurée sur la Fig.8. Cependant, toute la zone de production n'est pas intéressante. Entre le point d'origine des axes de coordonnées $(0, 0)$ où la production est nulle et le point A, il existe toute une série de combinaisons de productions des deux biens. Pour déterminer la meilleure situation, il faut se doter d'un critère. Il faut en effet considérer les prix de vente p_1 et p_2, les coûts variables v_1 et v_2 ainsi que le coût fixe F. Ces

1971).

[44]- Citons à ce propos R.A.FRISCH (*Maxima et minima - Théorie et applications économiques*, Paris, Dunod, Coll. Finance et économie appliquée, 178 p., 1960) pour une présentation de la problématique des calculs d'optimisation par rapport à la théorie économique, ainsi que L.V.KANTOROVITCH (*Calcul économique et utilisation optimale des ressources*, Paris, Dunod, 1959).) à propos de la technique de programmation proprement-dite. Voir D.LACAZE (1990, pp.78-84 ainsi que pp.188-94). Les modèles macroéconomiques développés par l'INSEE ont d'ailleurs pu être utilisés en "mode optimisation" - voir à ce propos les comptes PY PZ (Commissariat Général du Plan, *Défis à l'économie française*, Etudes et recherches, N°3-4, 1986, pp.19-47).

[45]- Voir à ce propos D.LACAZE ("La détermination de prix fictifs pour l'évaluation des projets - optimisation du système productif et calcul de prix fictifs", *Annales d'économie et statistique*, N°17, 1990).

variables peuvent être reliées dans une fonction objectif de Profit tel que :

$$\Pi = \sum_{i=1}^{N}(p_i - v_i) - F$$

Ainsi, la solution à notre problème de gestion de production consiste à déterminer la valeur maximale suivante :

$$Max\{(p_1 - v_1).x_1 + (p_2 - v_2).x_2 - F\}$$
$$\left|\begin{array}{l} a_1^1.x_1 + a_2^1.x_2 \leq a_{max}^1 \\ a_1^2.x_1 + a_2^2.x_2 \leq a_{max}^2 \\ x_1 \leq 0 \\ x_2 \leq 0 \end{array}\right.$$

Graphiquement, le problème revient à "caler" la droite d'isomarge[46] (fonction objectif) au point le plus haut (mode maximisation) du domaine de production réalisable, ce domaine est délimité par des arêtes. Et la rencontre entre deux arêtes s'appelle un sommet.

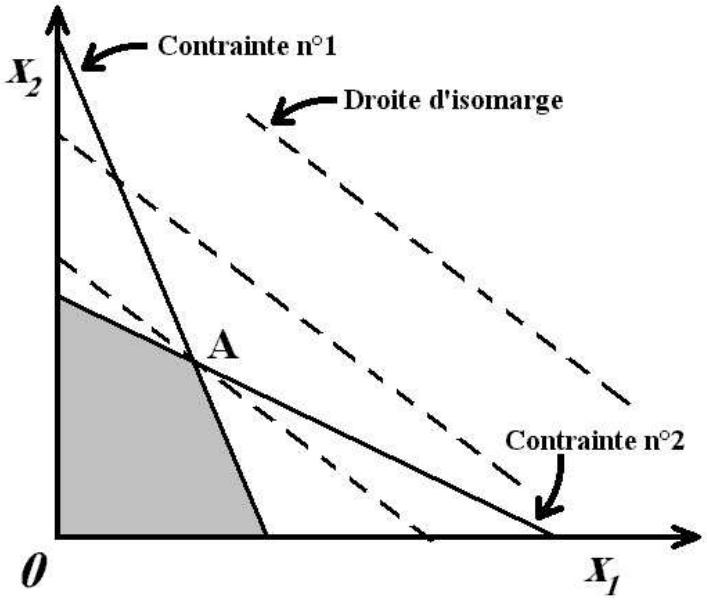


Fig.8 - Résolution graphique d'un programme linéaire

[46]- On peut en effet raisonner sur la maximisation de la marge et non plus sur la maximisation du profit en supprimant F qui ne dépend pas des quantités de marchandises.

L'algorithme du Simplexe (dite de Dantzig)

La méthode de résolution souvent utilisée est celle du simplexe[47]. Cependant des méthodes basées sur les mêmes principes que la résolution par élimination des systèmes d'inéquations avaient déjà été mises au point[48]. Leur maniement étant délicat, G.B.DANTZIG leva la difficulté modifiant ces inéquations de la manière suivante : $\forall i \in [1, N]$ et $\forall j \in [1, N]$

$$\begin{array}{l} Max\{f(x_i)\} \\ \left| \begin{array}{l} \sum_{i=1}^{M} a_i.x_i \leq b_j \\ \\ x_i \leq 0 \end{array} \right. \end{array}$$

devient

$$\begin{array}{l} Max\{f(x_i)\} \\ \left| \begin{array}{l} \sum_{i=1}^{M} a_i.x_i + x_{M+i} = b_j \\ \\ x_i \leq 0 \end{array} \right. \end{array}$$

où les x_{M+i} sont des variables artificielles, les "variables d'écart"[49]. L'algorithme démarre d'abord d'une solution réalisable du système - en général la solution nulle - *i.e.* la production nulle. Du fait que le système comporte des variables d'écart, il est surdéterminé. Cela implique qu'à chaque étape de calculs de l'algorithme, les coordonnées de la solution sont exprimées dans un nombre restreint de variables (variables de base) les autres étant nulles (variables hors base). A

[47]- Pour consulter des programmes en FORTRAN voir J.F. PHÉLIZON (1976, tome 1, pp.63-71). On pourra également consulter les travaux relatifs au premier codage des algorithmes de programmation linéaire in W.ORCHARD-HAYS (1955) ainsi que W.ORCHARD-HAYS et al. (1956).

[48]- G.B.DANTZIG (1966, pp.55-60) rappelle en effet que le mathématicien français J.B.J.FOURIER (1826) puis T.S.MOTZKIN (1936) avaient travaillé chacun en leur temps à la mise au point d'une méthode d'élimination pour les systèmes d'inéquations.

[49]- En principe, l'algorithme du Simplexe requiert des variables non négatives et un sens déterminé des inégalités, mais l'usage des variables d'écart permet de contourner certains obstacles. On pose en effet $x_1 = x_i^{(1)} + x_i^{(2)}$ avec $x_i^{(1)} \geq 0$ et $x_i^{(2)} \geq 0$. De même le sens des inégalités n'est pas un problème :

$$\begin{array}{lcl} \sum_{i=1}^{M} a_i.x_i \leq 0 & \text{devient} & \sum_{i=1}^{M} a_i.x_i + X_{M+i} = 0 \\ & \text{et} & \\ \sum_{i=1}^{M} a_i.x_i \geq 0 & \text{devient} & \sum_{i=1}^{M} a_i.x_i - X_{M+i} = 0 \end{array}$$

avec $x_{M+i} \geq 0$ et $i \in [1, M]$. Cela étant, G.B.DANTZIG & W.ORCHARD-HAYS (1953) ont proposé une variante appelée forme produit de l'inverse qui permet de résoudre les problèmes d'optimisation quelles que soient les données initiales - voir à ce propos G.B.DANTZIG (*op.cit.*, p.163).

partir de ces coordonnées, on calcule la valeur de la fonction objectif. Puis, on se déplace sur une autre "arête" du domaine en changeant de base - *i.e.* en exprimant les variables de base en variables hors base - par une technique de pivot. La fonction objectif étant exprimée selon des variables d'écart, on a atteint l'optimum lorsque ces dernières contribuent de manière négative à l'accroissement de la fonction objectif. Deux écueils peuvent survenir lors de la résolution : 1 - l'existence d'une infinité de solutions (en raison des sens incompatibles des inégalités, le domaine n'est pas borné) et 2 - le cas de "dégénérescence" (la fonction objectif est parallèle à l'une, au moins, des contraintes).

La programmation en nombres entiers et la programmation mixte

Dans certains cas, le problème à résoudre exige une solution en nombres entiers, tout simplement parce qu'une solution fractionnaire n'aurait aucun sens. Par exemple, on implante un nouvel atelier de production ou bien on en implante aucun, mais certainement pas un "demi-atelier". Nous n'entrerons pas dans le détail des algorithmes de programmation en nombres entiers ni de ceux de la programmation mixte (coexistence de variables entières et réelles)[50]. Nous préciserons seulement qu'elles s'articulent sur un algorithme du simplexe associé à une arborescence de contraintes logiques. Ces dernières contraintes s'expriment par des relations entre des variables du type x_i, x_j etc. (où, par exemple, i, j etc. sont des projets) et renvoient à des relations d'incompatiblité (ex. : $x_i + x_j \leq 1$), d'alternative exclusive (ex. : $x_i + x_j = 1$) ou non exclusive (ex. : $x_i + x_j \geq 1$) et d'implication (ex. : $x_i \leq x_j$)[51]. A partir de toutes les combinaisons possibles (et réalistes) des variables booléennes (*i.e.* logique), on forme une arborescence. La résolution du programme consiste alors à se "déplacer" sur l'arborescence jusqu'à obtenir la valeur optimale du programme, si au moins l'une des variables booléenne est non nulle.

ii - La programmation non linéaire

La formalisation des phénomènes économiques sous forme linéaire est une configuration bien commode, mais qu'il n'est malheureusement pas toujours possible d'obtenir. Dans le cas non linéaire, des difficultés surviennent cette fois dans la mesure où les algorithmes ne peuvent plus se "déplacer" sur des sommets (puisqu'il n'y a plus de sommet), où les solutions optimales peuvent être à l'intérieur du domaine réalisable (et non plus à la frontière). De plus, les algorithmes risquent de sortir du domaine pendant les calculs. Nous aborderons ici les deux principales configurations qui intéressent les modélisateurs qui doivent résoudre des programmes non linéaires, à savoir les problèmes avec et ceux sans contraintes (celles-ci étant linéaires ou non, sous forme d'équations ou d'inéquations).

[50]- Voir à ce propos G.B.DANTZIG (*op.cit.*, pp.332-70), D.LACAZE (*op.cit.*, 1990, pp.85-96) pour une présentation avec des exemples. Pour des programmes voir J.F.PHÉLIZON (*op.cit.*, tome 1, pp.73-90).

[51]- En l'occurrence, cela signifie que la réalisation du projet i est subordonnée à celle de j.

Les algorithmes de l'optimisation sans contrainte

Les deux algorithmes généralement cités permettant de résoudre ce type de problèmes sont les Méthodes du Gradient, d'une part et la Méthode de Newton-Raphson.

Méthode du gradient (dite de plus grande pente, dite de Cauchy[52])

Cette méthode, qui peut également être classée dans les méthodes de résolutions de systèmes d'équations non linéaires, en tant que méthode itérative non stationnaire (D.DUREISSEIX, *op.cit.*, pp.26-30), consiste à se déplacer d'un point $x^{(i)}$ à un autre $x^{(i+1)}$, dans le domaine de réalisation selon la direction donnée par le vecteur $\vec{\nabla} f(x^i)$ - où $\vec{\nabla}$ est le vecteur gradient, f la fonction objectif et la valeur de x à $i$ème itération. On se déplace dans cette direction jusqu'à ce que la fonction objectif soit à sa valeur maximale, *i.e.* au point suivant $x^{(i+1)}$. En ce point on détermine le gradient et ainsi de suite. La méthode du gradient classique qui peut se formuler de la manière suivante, $\vec{x}^{(i+1)} = \vec{x}^{(i)} + t^{(i)}.\vec{\nabla} f(\vec{x}^{(i)})$, ne converge pas très rapidement en raison du pas $t^{(i)}$. C'est pourquoi R.M. HESTENES et E. STIEFEL (1952) ont proposé la Méthode du gradient conjugué où le choix du pas est optimisé. Quoiqu'il en soit, cette méthode reste limitée dans la mesure où elle présente le risque de ne fournir qu'un optimum local.

Méthode de Newton-Raphson[53]

Plus utilisée en raison de sa convergence plus rapide et de sa capacité à déceler les optima globaux, cette méthode (I.NEWTON, 1671 ; J.RAPHSON, 1690), consiste à itérer à partir de l'équation suivante :

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} + \vec{\nabla} f(\vec{x}^{(i)}).H^{-1}(\vec{x}^{(i)})$$

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} + t^{(i)}.\vec{\nabla} f(\vec{x}^{(i)}).H^{-1}(\vec{x}^{(i)})$$

selon les propriétés de convergence de l'algorithme, avec H est la matrice hessienne supposée inversible

$$H = \frac{\delta^2 f}{\delta x_p \delta x_q}(\vec{x}^{(i)})$$

avec $p \in [1, N]$ et $q \in [1, N]$.

[52]- A propos des Méthodes du gradient, voir P.LASCAUX et R.THÉODOR (*op.cit.*, tome 2, pp.489-554), ainsi que M.SIBONY et J.C.MARDON (*op.cit.*, tome 1, pp.II-30). Pour une présentation avec organigramme, voir M.SIBONY et J.C.MARDON (*op.cit.*, tome 1, 1982, pp.II-126-38) ; pour une présentation avec organigramme et programmes FORTRAN, voir J.VIGNES (1980, pp.117-43).

[53]- Voir à ce propos les développements de M.SIBONY et J.C.MARDON (*op.cit.*, tome 1, 1982, pp.II-54- suiv.).

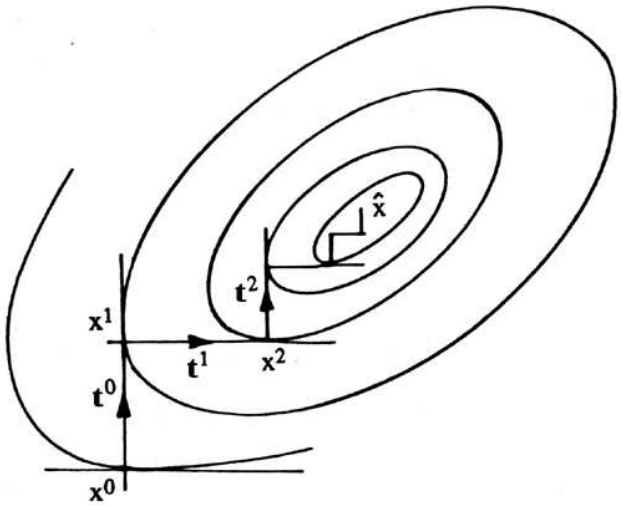


Fig.9 - Convergence en "zigzag" de la méthode du gradient[54]

Dans la pratique, comme pour les algorithmes de relaxation, les algorithmes ne s'arrêtent quasiment jamais en atteignant la valeur zéro. On doit se fixer un seuil de convergence ε qui va dépendre de la précision permise par les données. En tout état de cause, le principe général de la méthode du gradient laisse de nombreuses possibilités de relaxation[55] notamment en le combinant à la technique du simplexe (P.WOLFE, 1959). Cette méthode est utilisée pour programmer les algorithmes d'anticipations rationnelles[56].

Les algorithmes de l'optimisation sous contraintes

La méthode classique de résolution des programmes d'optimisation sous contraintes a été proposée par J.L.LAGRANGE. C'est la techniques dite des Multiplicateurs de Lagrange. Cependant, les conditions de validités de cette techniques sont limitées (H.W.KUHN, A.W.TUCKER, "Nonlinear Programming", *Econometrica* , N°19(1), jan., 1950, pp.50-51). C'est pourquoi d'autres algorithmes ont été proposés, tels que ceux de E.M.L.BEALE (1959), de DANTZIG ou de WOLFE.

La méthode du multiplicateur de Lagrange

L'approche marginaliste situe le comportement des agents, en particulier le consommateur, dans une problématique de maximisation de son utilité[57]. Ainsi,

[54]- D'après D.LACAZE (*op.cit.*, p.185).

[55]- Pour une présentation générale voir D.LACAZE (*op.cit.*, pp.183-88) ainsi que J.L.LIONS et G.I.MARCHOUK (EDS) (1974, pp.235-49). Pour un exposé de la Méthode de VIGNES, voir l'auteur (*op.cit.*, pp.126-43).

[56]- Voir à ce propos M.JULLIARD ("DYNARE - A Program for the Resolution on Simulation of Dynamic Models with Forward Variables Through the Use of a Relaxation Algorithm", *Document de travail du CEPREMAP*, N°9612, mars, 1996, 13 p. + **Programmes en langage Gauss**).

[57]- D'ailleurs, la programmation est une manière alternative (et en même temps plus opérationelle) d'aborder les problèmes et les concepts marginalistes de la microéconomie : "*Qu'on se situe au niveau de la firme ou au niveau d'une économie, la théorie de la dualité fournit ainsi une expression rigoureuse du marginalisme*" (D.LACAZE, *op.cit.*, p.33).

son utilité U, fonction des quantités x_i des N marchandises peut être optimisée sous contrainte de son revenu R, exprimé en termes d'achat des x_i marchandises

$$R = \sum_{i=1}^{N} p_i.x_i$$

où p_i est le prix de la i-ème marchandise. Si U et R sont définies dans l'espace des marchandises, on forme le "programme d'optimisation de Lagrange" suivant[58]

$$\begin{array}{l} Max\{U(x_1, x_2)\} \\ \Big| \\ R = p_1.x_1 + p_2.x_2 \end{array}$$

On pose $L'_{x_1}(x_1, x_2) = U(x_1, x_2) - \lambda(R - p_1.x_1 - p_2.x_2)$
puis on calcule

$$L'_{x_1}(x_1^*, x_2^*) = U'_{x_1}(x_1, x_2) - \lambda.p_1$$

et

$$L'_{x_2}(x_1^*, x_2^*) = U'_{x_2}(x_1, x_2) - \lambda.p_2$$

avec

$$R = p_1.x_1 + p_2.x_2 = 0$$

Graphiquement - Fig.10 - cela revient à effectuer une translation des fonctions U jusqu'à obtenir un point de tangence (x_1^*, x_2^*) avec la contrainte (ici la droite de budget du consommateur).

Cette technique a été amendée par H.W.KUHN et A.W.TUCKER (*op.cit.*) qui ont établi les conditions d'optimalité suivantes sur les multiplicateurs de Lagrange :

$$\left(\begin{array}{ccc} \frac{\delta L}{\delta x_i} \leq 0 & x_i \geq 0 & x_i.\frac{\delta L}{\delta x_i} = 0 \\ \\ \frac{\delta L}{\delta \lambda_i} \leq 0 & \lambda_i \geq 0 & \lambda_i.\frac{\delta L}{\delta \lambda_i} = 0 \end{array} \right)$$

[58]- Pour une présentation pédagogique, on pourra consulter à ce propos des manuels de microéconomie tels que B.GUERRIEN et B.NEZEYS (Microéconomie et calcul économique, Paris, Economica, 1982, pp.17-24).A propos des liens entre méthode du Simplexe et Multiplicateur de Lagrange, voir notamment G.B.DANTZIG (*op.cit.*, 1966, pp.108-113). On trouvera dans R.A.FRISCH (*op.cit.*, pp.39-48) une présentation qui permet d'apprécier le passage de l'intuition à la formalisation.

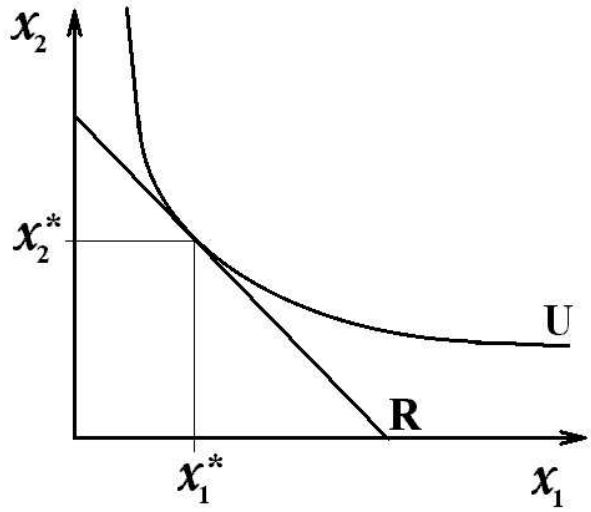


Fig.10 - Maximisation d'une fonction d'utilité dans un espace à deux marchandises

Les méthodes de programmation convexe de Beale, Dantzig et Rosen

Les méthodes de Beale et de Dantzig proposent de traiter les programmes convexes selon un algorithme proche de celui du simplexe. On introduit des variables d'écarts pour ne plus avoir à traiter d'inégalités. Dans la Méthode de Beale, le critère d'entrée ou de sortie de la base est cette fois déterminé par annulation des dérivées partielles de la fonction objectif alors que la Méthode de Dantzig, examine la compatibilité entre variables primales et variables duales[59]. La Méthode de Rosen[60] reprend la Méthode du gradient en introduisant une phase de vérification des inéquations relatives aux contraintes ($\lambda_j \geq 0$) à chaque

[59]- On peut en effet montrer que le programme primal suivant

$$Max\{Z_1 = \sum_{i=1}^{M} c_i.x_i\} \left| \begin{array}{l} \sum_{i=1}^{M} c_i.x_i \leq b_j \\ x_i \geq 0 \end{array} \right.$$

est équivalent à son dual :

$$Min\{Z_2 = \sum_{j=1}^{N} b_j.x_j\} \left| \begin{array}{l} \sum_{j=1}^{N} b_j.x_j \leq c_i \\ x_j \geq 0 \end{array} \right.$$

[60]- Voir J.B.ROSEN, "Gradient Projection Method for Non-Linear Programming - Part 1, Linear Constraints", *J. Soc. Industr. Appl. Math.*, N°8(1), 1960, pp.181-217, "Gradient Projection Method for Non-Linear Programming - Part 2, Non-Linear Constraints", *J. Soc. Industr. Appl. Math.*, N°9(4), 1961, pp.514-32.

itération. L'algorithme s'arrête soit à la frontière du domaine soit à l'optimum de la fonction objectif.

Le contrôle optimal

Cette technique[61] consiste à formaliser un processus dont on souhaite contrôler l'évolution en l'exprimant par une fonction continue du temps. On considère un système dont l'état est représenté par les vecteurs

$$x_t = \Big(x_1(t), x_2(t), \dots x_n(t) \Big)$$

et

$$u_t = \Big(u_1(t), u_2(t), \dots u_m(t) \Big)$$

ce dernier étant un vecteur de commandes qui détermine l'état de x. Les variations de la variable x sont données par le système d'équations différentielles :

$$\frac{dx_i}{dt} = f_i(x_1, \dots x_N, u_1, \dots u_M)$$

$\forall i \in [1, N]$. Les contraintes sur la variable d'état et de commande sont de la forme : $g_h\big(x(t), u(t), t\big) \geq 0 \; \forall t \in [t_0, t_h]$ et $h \in [1, q]$. Au moment de la résolution, on peut imposer non seulement des conditions initiales $x_{t_0} = x_0$ (indispensable pour la résolution différentielle), mais également des conditions de fin de période $x_{t_1} = x_1$ - pour des développements plus importants voir D.LACAZE (*op.cit.*, pp.195-236).

iii - Les principales applications de la recherche opérationnelle

La plupart des applications de recherche opérationnelle se basent sur l'algorithme du Simplexe. Cependant, d'autres concepts interviennent dans la recherche opérationnelle, l'arbre et le graphe, dans la formalisation des problèmes de transport, d'approvisionnement, d'ordonnancement ainsi que pour la programmation dynamique.

Les notions d'arbre et de graphe

Un outil important, que nous avons seulement évoqué dans les développements précédants, est utilisé en recherche opérationnelle ; il s'agit du graphe. Un graphe est défini (D.LACAZE, *op.cit.*, p.98) par la donnée : d'un ensemble X (individus, localités, opérations etc.) ; d'un ensemble U de couples ordonnés $(a, b)/a \in X$ et $b \in X$. Les éléments de X seront représentés par des points appelés sommets du graphe. Chaque élément (a, b) de U sera représenté par un arc fléché joignant d'extrémité initiale a à l'extrémité terminale b.

Ce type d'outil est particulièrement intéressant pour la résolution des problèmes de transports, d'affectation ou d'ordonnancement. La possibilité d'associer une matrice à un graphe ouvre en effet des perspectives de résolution

[61]- Conçue au départ pour contôler la trajectoire des vaissaux spatiaux soviétiques - voir L. PONTRYAGIN, V. BOLTYANSKII, R. GAMKRELIDZE et E. MISHCHENKO, *The Mathematical Theory of Optimal Processes*, Pergamon Press, 1964.

algébrique à des problèmes seulement schématisés graphiquement. On peut définir des configurations particulières de graphes (circuit, chaîne, cycle, chemin, etc.) ou des éléments du graphes ayant des propriétés spécifiques (noyau)[62].

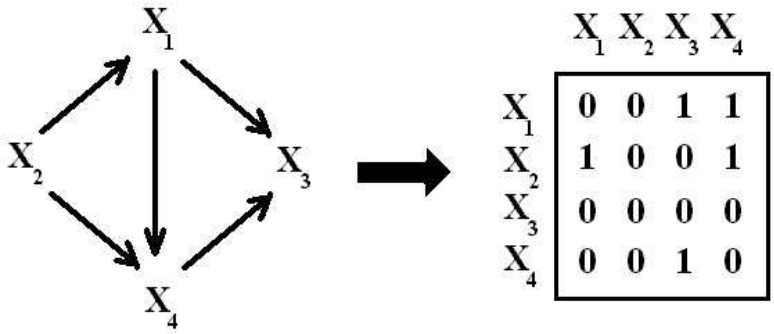


Fig.11 - Graphe et matrice associée

Enfin, on peut enrichir les processus en y introduisant des probabilités (chaîne de Markov - pour un développement mathématique et des programmes en FORTRAN voir J.F.PHÉLIZON, *op.cit.*, tome 2, pp.253-311).

Problème de transport

Le recourt au graphe est tout à fait approprié pour l'étude des problèmes de réseaux de transport (ils représentent des noeuds de communication, des capacités de circulation etc.). Un problème courant consiste à déterminer le flux maximum admis par le réseau. De type de problème peut être résolu en utilisant un simplexe du type :

$$Max\{\phi(\bar{b},\bar{a})\}$$

$$\left|\begin{array}{l} b(a_i,a_j) \leq \phi(a_i,a_j) \leq c(a_i,a_j) \\ \\ \displaystyle\sum_{a_k \in P(a_i)} \phi(a_k,aj) = \sum_{a_j \in S(a_i)} \phi(a_i,aj) \end{array}\right.$$

où $\phi(\bar{b},\bar{a})$ est le flux entre a et b, $b(a_i,a_j)$ et $c(a_i,a_j)$ (resp.) sont les capacités minimale et maximale (resp.) de flux et enfin, où P et S (resp.) sont les ensembles des prédecesseurs et des successeurs de a_i (D.LACAZE, *op.cit.*, pp.99-112). Néanmoins il existe des algorithmes spécifiques permettant de gagner du temps, notamment en termes de formalisation. L'Algorithme de Ford-Fulkerson (1962) fonctionne le long d'un chemin, par marquage des étapes permettant d'augmenter la valeur d'une chaîne (J.F.PHÉLIZON, *op.cit.*, tome 1, pp.131-183). L'Algorithme de Balas-Hammer, qui s'attache davantage au problème du coût de transport, opère par saturations successives des destinations du réseau (*Ibid.*). Enfin, l'Algorithme hongrois - d'après les travaux de J.ERGÉVARY

[62]- Voir J.F.PHÉLIZON (*op.cit.*, tome 1, pp.2-36) à propos de la théorie des graphes et des programmes en FORTRAN correspondants.

(1931) et G. KÖNIG (1936)[63] - a été développé pour traiter les problèmes d'affectation.

Ordonnancement

Etant donné le découpage d'un projet en tâches élémentaires dont on connaît la durée et les antériorités (*i.e.* la tâche j ne démarre que lorsque la tâche i est achevée), on cherche à savoir quelle sera la durée minimale d'exécution du projet[64]. Comme pour la programmation en nombres entiers, on formalise de manière mathématique des contraintes qualitatives : 1 - contraintes potentielles : $t_i \leq \mu + d_i$ (la tâche datée en t_i ne peut avoir lieu après une date μ avec une marge d_i autour de μ ; 2 - contraintes de succession et contraintes disjonctives : $t_i \geq t_j + k.\alpha_{ij}$ où i ne peut avoir lieu avant j plus ou moins une marge α_{ij} (deux tâches peuvent aussi s'exclure mutuellement dans le temps, parce qu'elles mobilisent les mêmes ressources). Le projet est alors représenté par un graphe. Chaque arc représente une tâche élémentaire du projet et se voit affecté d'une valeur correspondant à la durée de la tâche (il y a cependant des petites variantes de représentation entre la méthode PERT et la méthode des potentiels). Ces algorithmes visent à calculer le chemin minimal de développement du projet. On trouvera des développements de ces algorithmes tels que, la Méthode Ford, la Méthode de Bellman-Kalaba, la Méthode matricielle ou la Méthode PERT notamment dans D.LACAZE (1990, *op.cit.*, pp.113-27), J.F.PHÉLIZON (*op.cit.*, tome 1, pp.93-128 avec des programmes en FORTRAN) ainsi que R.FAURE (*op.cit.*) et G.LÉVY (1994, pp.115-60 avec des programmes en Pascal). Ces algorithmes parcourent de manière itérative l'ensemble du réseau, en gardant en mémoire les valeurs temporairement minimales et les étapes correspondantes. Cependant, un algorithme basé sur le calcul matriciel de réseaux (Réseaux de Pétri) a été développé plus récemment[65].

Gestion de stocks et des files d'attente

Le problème de l'adéquation en temps réel entre une demande - qui se manifeste par l'arrivée de clients dont le nombre peut être probabilisé - et une offre - qui se manifeste par l'état du stock de l'entreprise - microéconomiques, a été résolu au moyen d'algorithme de gestion des stocks et de simulation de file d'attente (D.LACAZE, 1990, *op.cit.*, pp.133-59). Partant de l'hypothèse d'écoulement linéaire des stocks, on peut représenter le déstockage du magasin de l'entreprise de la manière suivante - voir Fig.12. Si S est l'état du stock, t le temps alors si S s'écoule linéairement et que l'on connaît d, le délai de livraison, on est capable de déterminer par projection la quantité S_c correspondant à la date de réapprovisionnement au plus tard - c'est la Méthode du point de commande.

[63]- Cités dans la bibliographie d'A.KAUFMANN (1970, tome 2) Voir également le tome 2. Pour des programmes en Pascal, voir G.LÉVY (*op.cit.*).

[64]- Les premières méthodes ont été proposées par le Department of the Navy (1958) et par la NASA (*Apollo Configuration Management Manual*, NPC 500-1, Washington, Office of Manned Space Flight, 1964).

[65]- J.PETERSON, *Petri Net Theory and the Modelling of Systems, Englewood Cliffs*, Prentice Hall, 1981.

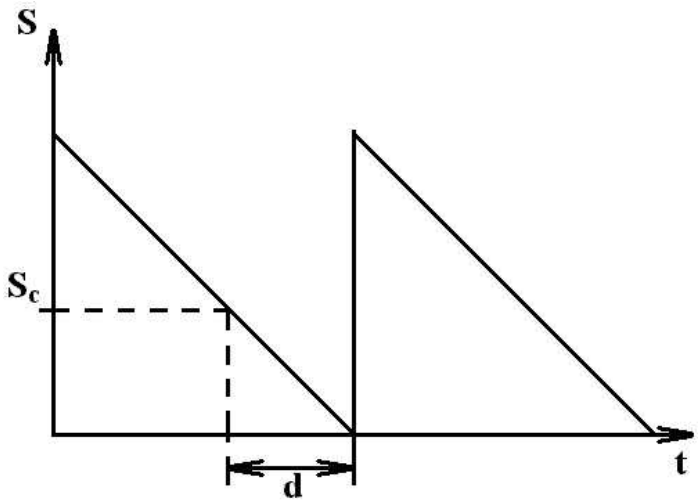


Fig.12 - Méthode du point de commande

Cependant le flux de demande peut être moins simple et nécessiter des simulations spécifiques ; en introduisant des aléas. Par exemple, le nombre probable de clients peut varier fortement d'une période à l'autre sur une échelle de temps jugée pertinente par l'entreprise (la journée, la semaine, le mois). Dans ce cas là, on peut simuler les arrivées et des sorties par la méthode de Monte-Carlo, ce qui permet de calculer le temps moyen d'attente aux caisses (J.F.PHÉLIZON, *op.cit.*, tome 2, pp.197-250 ainsi que R.FAURE *op.cit.*)[66].

Programmation dynamique

Il s'agit cette fois de trouver des solutions optimales (à tout le moins, satisfaisantes) à chaque période de développement d'un processus économique ou de gestion. Contrairement au contrôle optimal, on raisonne ici plutôt en temps discret. L'Algorithme de Bellman (1957) repose sur une série de programmes d'optimisation analogues à ceux de l'optimisation de flots dans un réseau.

Jeux

On ne peut guère parler de recherche opérationnelle sans évoquer la Théorie de jeux. Celle-ci propose de déterminer les conditions d'équilibre entre différents joueurs d'une partie (guerre, concurrence sur un marché etc.). Elle ne se place donc pas dans l'optique d'un des joueurs, mais dans l'optique générale de la partie, chaque joueur usant au mieux de ses propres atouts. Dans les jeux de décision tels que la concurrence sur un marché (il existe plusieurs types de jeux : d'information, de décision, contre la nature, etc.), on peut représenter le résultat de la confrontation entre deux joueurs A et B grâce à une matrice G, dite matrice des gains :

$$G = \begin{pmatrix} U_A^{1,1}.U_B^{1,1} & & \\ & U_A^{i,j}.U_B^{i,j} & \\ & & U_A^{n,n}.U_B^{n,n} \end{pmatrix}$$

[66]- On trouvera en Annexe 1.1, à titre d'illustration d'offre de transport urbain, le recourt aux fonctions *modulo* que nous avons proposé pour modéliser un trafic alterné sur une ligne de métro avec fourches. A propos de l'usage des nombres *modulo* en gestion, voir A.KAUFMANN et A.HENRY-LABORDIERE (1974, pp.387-91).

$\forall i \in [1, n]$. Les variables utilisées sont n, le nombre de décisions différentes possibles, $U_A^{i,j}$ l'utilité retirée par A sachant que A a pris la décision i et B la décision j. Les outils dont nous avons déjà parlé jusqu'à présent en optimisation linéaire (graphes, réseaux, etc.) permettent alors de mettre en lumière les stratégies les plus pertinentes compte tenu des paramètres des différents joueurs[67]. Il faut toutefois préciser que la Théorie des jeux utilise assez peu les techniques de simulation. Dans ces rares cas, notamment les jeux d'entreprises programmés, il n'y a alors pas d'algorithmes réellement spécifiques[68]. En revanche, les spécialistes des jeux utilisent beaucoup l'outil mathématique à des fins purement théoriques et les validations, lorsqu'elles existent, seraient plutôt du domaine de l'économie expérimentale[69].

RÉFÉRENCES

AMSTUTZ A.E., (1967), *Computer Simulation of Competitive Market Response*, Cambridge (Mass.), MIT Press, 457 p.

ANDERSON E., BAI Z., BISCHOF C., BLACKFORD S., DEMMEL J., DON-GARRA J., DU CROZ J., GREENBAUM A., HAMMERLING S., McKENNEY A., SORENSEN D., (1999), *LAPACK Users' Guide*, Philadelphia, PA, Society for Industrial and Applied Mathematics + Programmes.

BEALE E.M.L., (1959), "On Quadratic Programming", *Naval Research Logisitic Quarterly*, N°6(3), sept., pp.227-43.

BELLMAN R.E., (1957), *Dynamic Programming*, Princeton, Princeton UP.

BENOIT (Cdt), (1924), "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues - application de la méthode à la résolution d'un système d'équations linéaires (procédé du Commandant Cholesky)", *Bulletin Géodésique*, N°2, 1924, pp. 5-77.

BERSTEL J., PIN J.E., POCCHIOLA M., (1991), *Mathématiques et informatique - tome 1 Algèbre, Paris, Ediscience international*, Coll. Informatique, 245 p. + Programmes.

BOIS F.Y., MASZLE D.R., (1997), *MCSim : A Monte Carlo Simulation Program*, Boston, Free Software Foundation, 60 p. + Programmes.

CAUCHY A.L., (1847), "Méthode générale pour la résolution des systèmes d'équations simultanées", *C.R.Acad. Sc.*, N°25, pp.536-38.

CORSIM, (2000), *CORSIM 4.0 Programmer's Documentation (DRAFT)*, june, 187 p.

DANTZIG G.B., (1966), *Linear Programming and Extensions*, Princeton, Princeton UP, (trad. Paris, Dunod), 433 p.

DANTZIG G.B., ORCHARD-HAYS W., (1953), "Notes on Linear Programming : Part V - Alternate Algorithm for the Revised Simplex Method Using Product Form of the Inverse", *RAND Corporation*, RM-1268, November 19.

DEPARTMENT OF THE NAVY, (1958), "PERT, Program Evaluation Research Task", *Phase I Summary Report*, Special Project Office, Washington, Bureau of Ordnance.

DION J.G., GAUDET R., (1996), *Méthodes d'analyse numérique - de la théorie à l'application*, Mont- Royal (Québec), Modulo, Coll. universitaire de mathématiques, 623 p.

DONY R., (1986), *Calcul des parties cachées - approximations des courbes par la méthode de Bezier et des B-Splines*, Paris, Masson, Coll. Méthodes+Programmes, 238 p.

ERGÉVARY J., (1931), "Matrixok kombinatorius tulajdonsagairol", *Mat. Fiz. Lapok.*, p.16.

[67]- Voir J.F.PHÉLIZON, *op.cit.*, tome 1, pp.187-221, pour des développements mathématiques et des programmes FORTRAN.

[68]- Les algorithmes reprennent les grandeurs de comptabilité générale ou de comptabilité analytique d'entreprise. Voir A.KAUFMANN et al. (1976, pp.84-118) ainsi que A.E.AMSTUTZ (1967, pp.89-110) à propos de la programmation des jeux d'entreprises (comportement de demande, de distribution etc., sur le marché) en langage FORTRAN, GPSS et SimScript.

[69]- Voir notre note (2000.a) au sujet de l'historique de l'économie expérimentale où nous précisons les liens entre psychologues, théoriciens des jeux et économiste expérimentaux dans les années 50-60 aux Etats-Unis et en Allemagne notamment.

FAURE R., (1979), *Précis de recherche opérationnelle*, Paris, Dunod, Coll. Décision, 466 p.

FEUVRIER C.V., (1971), *La simulation des systèmes - Maîtrise d'informatique*, Paris, Dunod, Coll. Université, 152 p.

FORD L.R., FULKERSON D.R., (1962), *Flows in Network*, Princeton, Princeton UP.

FOURER R., GAY D.M., KERNIGHAN B.W., (1990), "A Modeling Language for Mathematical Programming", *Management Science*, N°36, pp.519-54.

FOURIER J.B.J., (1826), "Solution d'une question particulière du calcul des inégalités", *Oeuvres II*, pp.317-28.

GAUSS J.K.F., (1826), "Theoria Combinationis Observationum Erroribus Minimis Obnoxiae", *Supplementum*, Vol.4, Göttingen, pp.55-93.

HERNERT P., (1995), *Les algorithmes*, Paris, PUF, Coll. Que sais-je ?, 128 p.

HESTENES M.R., STIEFEL E., (1952), "Methods of Conjugate Gradients for Solving Linear Systems", *J. Res. Natl.Bur. Stand.*, N°49, pp.409-36.

IFRAH G., (1981), *Histoire universelle des chiffres - l'intelligence des hommes racontée par les nombres et le calcul*, Paris, Laffont, Coll. Bouquins, (2 Vol.), 2050 p.

JACOBI C.G.J., (1846), "Über ein leichtes Verfahren, die in der theorie der säcularstörungen vorkommenden gleichungen numerish aufsulösen", *Journal für die reine und angewandle Mathematik*, pp. 51-94.

KAUFMANN A., (1968), *Méthodes et modèles de la recherche opérationnelle - tome 2*, Paris, Dunod, Coll. L'économie d'entreprise, 544 p.

KAUFMANN A., (1970), *Méthodes et modèles de la recherche opérationnelle - tome 1*, Paris, Dunod, Coll. L'économie d'entreprise, 536 p., (2nde éd.).

KAUFMANN A., FAURE R., LE GARFF A., (1976), *Les jeux d'entreprises*, Paris, PUF, Coll. Que sais-je ?, 128 p. (4ème éd.).

KAUFMANN A., HENRY-LABORDIERE A., (1974), *Méthodes et modèles de la recherche opérationnelle - tome 3*, Paris, Dunod, Coll. L'économie d'entreprise, 398 p.

KÖNIG G., (1936), *Theorie des Endlichen and Unendlichen Graphen*, Leipzig, Akad. Verl. MBH.

KNUTH D.E., (1981), *The Art of Programming - tome 2*, Seminumerical Algorithms, Reading (Mass.), Addison-Wesley, 688 p.

KUTTA W., (1901), "Beitrag zur naherungsweisen Integration totaler Differentialgleichunge", *Zeitung Mathematik Physik*, vol. 46, pp.435-53.

LA PORTE M., VIGNES J., (1980), *Algorithmes numériques, analyse et mise en oeuvre - Tome 2, équations et systèmes non linéaires*, Paris, Technip, Coll.Langages et algorithmes de l'informatique, 302 p.

LACAZE D., (1990), *Optimisation appliquée à la gestion et à l'économie*, Paris, Economica, 440 p.

LASCAUX P., THÉODOR R., (1986-87), *Analyse numérique matricielle appliquée à l'art de l'ingénieur (2 Vol.)*, Paris, Masson, 790 p.

LEEB H., (1995), *Random Numbers for Computer Simulation, Diplomarbeit zur Erlangung des Magistergrades an der Naturwissenschaftlichen Fakultät*, Salzburg, 1995, 135 p.

LÉVY G., (1994), *Algorithmique combinatoire - méthodes constructives*, Paris, Dunod, Coll.Science informatique, 502 p. + Programmes.

LIONS J.L., MARCHOUK G.I.(EDS), (1974), *Sur les méthodes numériques en sciences physiques et économiques*, Paris, Dunod, Coll. Méthodes mathématiques de l'informatique, 299 p.

MANAGEMENT SCIENCE SYSTEMS, (1970), *DATAFORM Mathematical Programming Data - User Manual*, Arlington, Management System Ketron.

MONASSE D., (1988), *Mathématiques et informatique*, Paris, Vuibert, Coll. Classes préparatoires Cours et travaux dirigés, 223 p. + Programmes.

MOTZKIN T.S., (1936), *Beitrage zur Theorie der Linearen Ungleichungen*, Doctoral Thesis, Universität Zürich.

NEWTON I, (1671), *Methodus Fluxionum et Serierum Infinitarium*, (éd., 1736 ; trad., 1740 ; réimp.1966).

ORCHARD-HAYS W., (1955), "RAND Code for Simplex", The RAND Corporation, *Research Memorandum*, RM 1440, February 7.

ORCHARD-HAYS W., CUTLER L., JUDD H., (1956), "Manual for the RAND IBM Code for Linear Programming on the 704", *The RAND Corporation*, Paper P-842, May 16, 24-26.

PHÉLIZON J.F., (1976), *Informatique opérationnelle - Tome 1, méthodes relevant de l'optimisation*, Paris, Economica, Coll. Informatique, 225 p.

PHÉLIZON J.F., (1977), *Informatique opérationnelle - Tome 2, modèles conduisant à la simulation*, Paris, Economica, Coll. Informatique, 355 p.

RAPHSON J., (1690), *Analysis Aequationum Universalis*, Londres.

REVERCHON A., DUCAMP M., (1993), *Outils mathématiques en C++*, Paris, A.Colin, Coll. U- informatique, 401 p. + Programmes.

RUNGE C., (1895), "Über die numerische Auflösungen von Differentialgleichungen", *Mathematische Annalen*, N°46, pp.167-78.

SCHRAGE L., (1989) *User's Manual for Linear, Integer and Quadratic Programming with LINDO*, Redwood City, Scientifc Press.

SEIDEL L., (1874), "Über ein Verfahren die Gleichungen, auf welche die Methode des kleinsten Quadrate führt, sowie lineäre Gleichungen überhaupt, durch successive Annäherung aufzulösen", *Communication à la section math-physique de l'Académie Royale des Sciences de Berlin*, séance du 7 fév.

SIBONY M., MARDON J.C., (1982), *Analyse numérique - Tome 1, systèmes linéaires et non linéaires*, Paris, Hermann, Coll.Actualité scientifique et industrielle. + Programmes.

SIBONY M., MARDON J.C., (1982), *Analyse numérique - Tome 2, approximations et équations différentielles*, Paris, Hermann, Coll.Actualité scientifique et industrielle. + Programmes.

SIBONY M., (1988), *Analyse numérique - Tome 3, itérations et approximations*, Paris, Hermann, Coll.Actualité scientifique et industrielle. + Programmes.

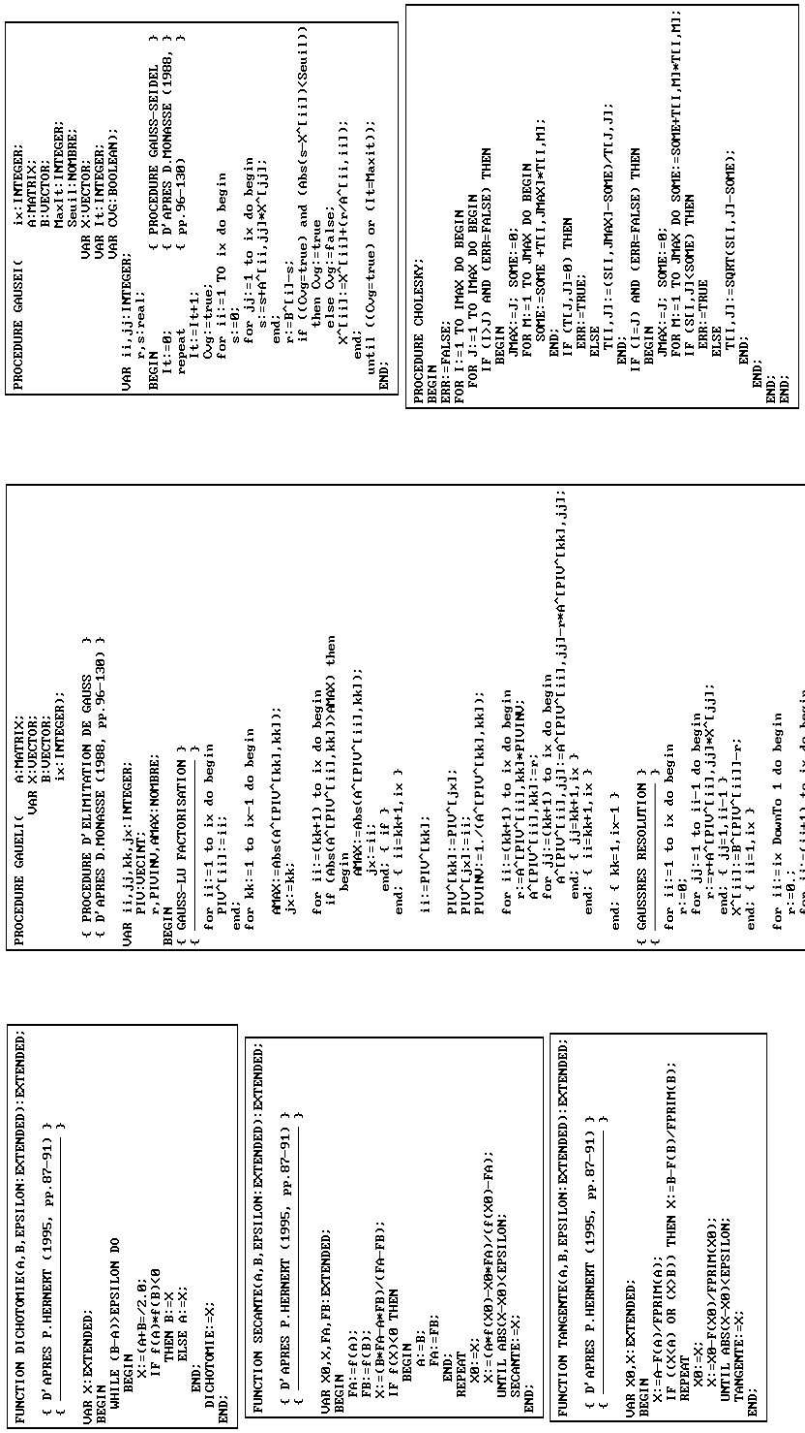
SOUTHWELL R., (1946), *Relaxation Methods in Theoretical Physics*, Oxford, Clarendon Press.

THÉODOR R., (1989), *Initiation à l'analyse numérique*, Paris, Masson, Coll.CNAM Cours A, 302 p.

VIGNES J., (1980), *Algorithmes numériques, analyse et mise en oeuvre - Tome 2, équations et systèmes non linéaires*, Paris, Technip, Coll.Langages et algorithmes de l'informatique, 302 p.

WOLFE P., (1959), "The Simplexe Method for Quadratic Programming", *Econometrica*, 27(3), jul.

ANNEXE 1.1 - PROGRAMMES D'ANALYSE NUMÉRIQUE ET DE SIMULATION



```
FUNCTION DICHOTOMIE(A,B,EPSILON:EXTENDED):EXTENDED;
{ D'APRES P.HERMENT (1995, pp.87-91) }
{ ____________________ }
VAR X:EXTENDED;
BEGIN
 WHILE (B-A)>EPSILON DO
   BEGIN
     X:=(A+B)/2.0;
     IF F(A)*F(X)<0
       THEN B:=X
       ELSE A:=X;
   END;
 DICHOTOMIE:=X;
END;

FUNCTION SECANTE(A,B,EPSILON:EXTENDED):EXTENDED;
{ D'APRES P.HERMENT (1995, pp.87-91) }
{ ____________________ }
VAR X0,X,FA,FB:EXTENDED;
BEGIN
  FA:=F(A);
  FB:=F(B);
  X:=(A*FB-B*FA)/(FB-FA);
  IF F(X)<0 THEN
    BEGIN
      A:=B;
      FA:=FB;
    END;
  REPEAT
    X0:=X;
    X:=(A*F(X0)-X0*FA)/(F(X0)-FA);
  UNTIL ABS(X-X0)<EPSILON;
  SECANTE:=X;
END;

FUNCTION TANGENTE(A,B,EPSILON:EXTENDED):EXTENDED;
{ D'APRES P.HERMENT (1995, pp.87-91) }
{ ____________________ }
VAR X0,X:EXTENDED;
BEGIN
  X:=A-F(A)/FPRIM(A);
  IF ((X<A) OR (X>B)) THEN X:=B-F(B)/FPRIM(B);
  REPEAT
    X0:=X;
    X:=X0-F(X0)/FPRIM(X0);
  UNTIL ABS(X-X0)<EPSILON;
  TANGENTE:=X;
END;

PROCEDURE GAUELI(    A:MATRIX;
                 VAR X:VECTOR;
                     B:VECTOR;
                     ix:INTEGER);
{ PROCEDURE D'ELIMITATION DE GAUSS    }
{ D'APRES D.MONASSE (1988, pp.96-130) }
VAR ii,jj,kk,jx:INTEGER;
    PIU:VECINT;
    r,PIVINV,AMAX:NOMBRE;
BEGIN
{ GAUSS-LU FACTORISATION }
{ ______________________ }
 for ii:=1 to ix do begin
   PIU^[ii]:=ii;
 end;
 for kk:=1 to ix-1 do begin

   AMAX:=abs(A^[PIU^[kk],kk]);
   jx:=kk;

   for ii:=(kk+1) to ix do begin
    if (abs(A^[PIU^[ii],kk])>AMAX) then
      begin
        AMAX:=abs(A^[PIU^[ii],kk]);
        jx:=ii;
      end; { if }
   end; { ii=kk+1,ix }

   ii:=PIU^[kk];

   PIU^[kk]:=PIU^[jx];
   PIU^[jx]:=ii;
   PIVINV:=1./(A^[PIU^[kk],kk]);

   for ii:=(kk+1) to ix do begin
     r:=A^[PIU^[ii],kk]*PIVINV;
     A^[PIU^[ii],kk]:=r;
     for jj:=(kk+1) to ix do begin
       A^[PIU^[ii],jj]:=A^[PIU^[ii],jj]-r*A^[PIU^[kk],jj];
     end; { jj=kk+1,ix }
   end; { ii=kk+1,ix }

 end; { kk=1,ix-1 }

{ GAUSSRES RESOLUTION }
{ ____________________ }
 for ii:=1 to ix do begin
   r:=0;
   for jj:=1 to ii-1 do begin
     r:=r+A^[PIU^[ii],jj]*X^[jj];
   end; { jj=1,ii-1 }
   X^[ii]:=B^[PIU^[ii]]-r;
 end; { ii=1,ix }

 for ii:=ix DownTo 1 do begin
   r:=0.;
   for jj:=(ii+1) to ix do begin
     r:=r+A^[PIU^[ii],jj]*X^[jj];
   end; { jj=ii+1,ix }
   X^[ii]:=(X^[ii]-r)/A^[PIU^[ii],ii];
 end; { ii=ix,1,-1 }

END;

PROCEDURE GAUSSEI(    ix:INTEGER;
                      A:MATRIX;
                      B:VECTOR;
                  VAR X:VECTOR;
                      Maxit:INTEGER;
                      Seuil:NOMBRE;
                  VAR It:INTEGER;
                  VAR CVG:BOOLEAN);
VAR ii,jj:INTEGER;
    r,s:real;
BEGIN                 { PROCEDURE GAUSS-SEIDEL   }
  It:=0;              { D'APRES D.MONASSE (1988, }
  repeat              { pp.96-130)               }
    It:=It+1;
    CVG:=true;
    for ii:=1 TO ix do begin
      s:=0;
      for jj:=1 to ix do begin
        s:=s+A^[ii,jj]*X^[jj];
      end;
      r:=B^[ii]-s;
      if ((CVG=true) and (abs(s-X^[ii])<Seuil))
        then CVG:=true
        else CVG:=false;
      X^[ii]:=X^[ii]+(r/A^[ii,ii]);
    end;
  until (((CVG=true) or (It=Maxit));
END;

PROCEDURE CHOLESKY;
BEGIN
ERR:=FALSE;
FOR I:=1 TO IMAX DO BEGIN
 FOR J:=1 TO IMAX DO BEGIN
  IF (I>J) AND (ERR=FALSE) THEN
   BEGIN
    JMAX:=J; SOME:=0;
    FOR M:=1 TO JMAX DO BEGIN
      SOME:=SOME +TI[I,JMAX]*TI[J,M];
    END;
    IF (TI[J,J]=0) THEN
     ERR:=TRUE;
    ELSE
     TI[I,J]:=(SI[I,J]-SOME)/TI[J,J];
   END;
  IF (I=J) AND (ERR=FALSE) THEN
   BEGIN
    JMAX:=J; SOME:=0;
    FOR M:=1 TO JMAX DO SOME:=SOME+TI[I,M]*TI[I,M];
    IF (SI[I,J]<SOME) THEN
      ERR:=TRUE
    ELSE
      TI[I,J]:=SQRT(SI[I,J]-SOME);
   END;
 END;
END;
END;
```

```
PROCEDURE GAUJOR(     B:MATRIX;
                  VAR X:VECTOR;
                      A:VECTOR;
                      N:INTEGER);
VAR I,J,K:INTEGER;
    L,M:VECINT;
    DET,H,PV:NOMBRE;
BEGIN                 { PROCEDURE GAUSS-JORDAN    }
  NEW(L);             { D'APRES NOTRE TRADUCTION  }
  NEW(M);             { DES PROGRAMMES FORTRAN    }
  DET:=1.;            { DE J.F.PHELIZON (1977,T2  }
  FOR K:=1 TO N DO    { pp.315-23)                }
    BEGIN
      L^[K]:=K;
      M^[K]:=K;
      PV:=B^[K,K];
      FOR I:=K TO N DO
        BEGIN
          FOR J:=K TO N DO
            BEGIN
              IF NOT(ABS(PV)>=ABS(B^[I,J])) THEN
                BEGIN
                  PV:=B^[I,J];
                  L^[K]:=J;
                  M^[K]:=I;
                END; { IF }
            END;
        END; { I=K,N / J=K,N / 15 CONTINUE }
      J:=L^[K];
      IF NOT(J<K) THEN
        BEGIN
          FOR I:=1 TO N DO
            BEGIN
              H:=-B^[I,K];
              B^[I,K]:=B^[I,J];
              B^[I,J]:=H;
            END; { IF }
        END; { I=1,N / 20 CONTINUE }
      I:=M^[K];
      IF NOT(I<K) THEN
        BEGIN
          FOR J:=1 TO N DO
            BEGIN
              H:=-B^[K,J];
              B^[K,J]:=B^[I,J];
              B^[I,J]:=H;
            END; { J=1,N / 30 CONTINUE }
        END; { IF }
      IF NOT(ABS(PV)>1.E-20) THEN
        BEGIN
          WRITE('LA MATRICE B EST SINGULIERE');
          HALT;
        END; { IF }
      J:=0;
      REPEAT
        J:=J+1;
      UNTIL ((J<N) THEN B^[K,J]:=-B^[K,J]/PV;
      I:=0;
      REPEAT
        I:=I+1;
        IF (I<>K) THEN
          BEGIN
            H:=B^[I,K];
            FOR J:=1 TO N DO
              IF ((I=N) OR (I=K) OR (J=K)) THEN
                B^[I,J]:=B^[I,J]+B^[L,K]*B^[K,J];
          END; { I=1,N / 60 CONTINUE }
      UNTIL ((I=N) OR (I=K));
      I:=0;
      REPEAT
        I:=I+1;
        IF (I<>K) THEN B^[I,K]:=B^[I,K]/PV;
      UNTIL ((I=N) OR (I=K));
      DET:=DET*PV;
      B^[K,K]:=1.0/PV;
    END; { K=1,N / 60 CONTINUE }
  K:=N;
  REPEAT
    K:=K-1;
    IF NOT(K<=0) THEN
      BEGIN
        I:=L^[K];
        IF NOT(I<=K) THEN
          BEGIN
            FOR J:=1 TO N DO
              BEGIN
                H:=B^[K,J];
                B^[K,J]:=B^[I,J];
                B^[I,J]:=H;
              END; { J=1,N / 70 CONTINUE }
          END; { IF }
        J:=M^[K];
        IF NOT(J<=K) THEN
          BEGIN
            FOR I:=1 TO N DO
              BEGIN
                H:=B^[I,K];
                B^[I,K]:=-B^[I,J];
                B^[I,J]:=H;
              END; { I=1,N / 80 CONTINUE }
          END;
      END; { IF }
  UNTIL (K<=0);
  FOR I:=1 TO N DO
    BEGIN
      X^[I]:=0.0;
      FOR J:=1 TO N DO
        BEGIN
          X^[I]:=X^[I]+B^[I,J]*A^[J];
        END;
    END; { I=1,N / J=1,N / 90 CONTINUE }
  DISPOSE(M);
  DISPOSE(L);
END;
```

```
FUNCTION F_PRIME(X:REAL) : REAL;
VAR H:REAL;
BEGIN
  H:=1.0E-08;
  FPRIM:=(F(X+H)-F(X))/H;
END;
```

```
FUNCTION F_SECONDE(X:REAL) : REAL;
VAR H:REAL;
BEGIN
  H:=1.0E-08;
  FSECOND:=(F(X+H)+F(X-H)-2.0*F(X))/H*2;
END;
```

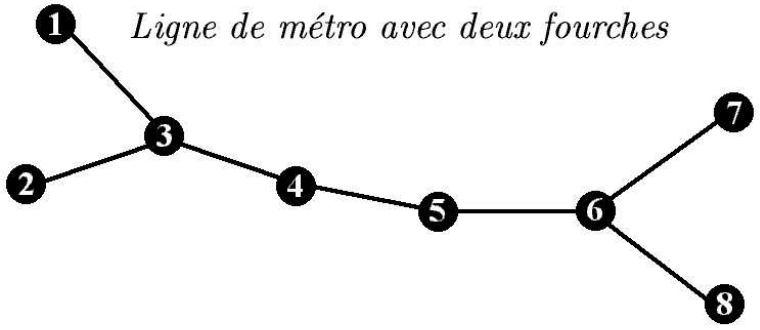
```
FUNCTION SIMPSON(G,B:EXTENDED;N:INTEGER):EXTENDED;
VAR DX,S:EXTENDED;
    I:INTEGER;
BEGIN                 { D'APRES D.MONASSE (1980,pp.159-73) }
  S:=0;               {  ET P.HENNERT (1995,pp.92-95)     }
  DX:=(B-A)/N;
  FOR I:=0 TO N-1 DO BEGIN
  S:=S+F(A+I*DX)+4*F(A+(2*I+1)*DX/2)+F(A+(I+1)*DX);
  END;
  SIMPSON:=S*DX/6;
END;
```

```
FUNCTION EVENEMENT(PROB:INTEGER) : BOOLEAN;
VAR TIRAGE:INTEGER;
BEGIN                 { EVENEMENT DONT LA PROBABILITE PROB }
  { AURA LIEU SI TIRAGE EST DANS LE BON  }
  { INTERVALLE C'EST-A-DIRE TIRAGE<=PROB }
  RANDOMIZE;
  TIRAGE:=RANDOM(100);
  IF (TIRAGE<=PROB) THEN EVENEMENT:=TRUE
                    ELSE EVENEMENT:=FALSE;
END;
```

```
Type donnees=Array[1..maxdegre1 of Real;
     LagrangePoly=Polynome;
Procedure Lagrange(     a,b:donnees;
                        nombre:integer;
                    Var c:LagrangePoly);

Var i,j:integer;    { D.MONASSE (pp.87-90) }
    d:donnees;
begin
  MetAZeroPoly(c);
  for i:=1 to Nombre do begin
    d[i]:=b[i];
    for j:=i-1 downto 0 do
      begin
        d[j]:=d[j+1]-d[j]/(a[i]-a[j]);
      end;
    c[i]:=d[0];
  end;
end;
Function EvalueLagrange(     c:LagrangePoly;
                             a:donnees;
                             x:real) : real;

Var i:integer;
    degre_c:integer;
    valeur:real;
begin
  degre_c=degre[c];
  valeur:=c[degre_c];
  for i:=degre_c-1 downto 0 do
    valeur:=valeur*(x-a[i])+c[i];
  EvalueLagrange:=valeur;
end;
```

Exemple de programme utilisant la fonction modulo : le trafic alterné

```
PROGRAM TRAFIC_ALTERNE;
VAR         F:ARRAY[1..3] OF INTEGER;
            I,J,N,NO:INTEGER;
            S:ARRAY[1..8] OF INTEGER;
            fx:Text;
BEGIN
F[1]:=1; F[2]:=2; F[3]:=4;
S[1]:=1; S[2]:=1; S[3]:=0; S[4]:=0;
S[5]:=0; S[6]:=0; S[7]:=1; S[8]:=1;
ASSIGN(fx,'METRO.OUT');
REWRITE(fx);
FOR J:=1 TO 3 DO BEGIN
WRITELN(fx,'┌──────┬──────────────────────┐');
WRITELN(fx,'│F=1|',F[j]:1,
'|    STATIONS     |');
WRITELN(fx,'├──────┼──┬──┬──┬──┬──┬──┬──┬──┤');
WRITELN(fx,'│TEMPS │1 │2 │3 │4 │5 │6 │7 │8 │');
WRITELN(fx,'├──────┼──┼──┼──┼──┼──┼──┼──┼──┤');
FOR N:=1 TO 20 DO
 BEGIN
  WRITE(fx,'|  ',N:2,' |');
  FOR I:=1 TO 8 DO
    BEGIN
{ 0 } IF (F[J]=0) THEN NO:=0;
{1|1} IF (F[J]=1) THEN
        BEGIN
          IF (S[I]=0) THEN
            BEGIN
              NO:=1;
            END
          ELSE
            BEGIN
              NO:=N MOD 2;
            END;
        END;
{1|2} IF (F[J]=2) THEN
        BEGIN
          IF (S[I]=0) THEN
            BEGIN
              NO:=N MOD 2;
            END
          ELSE
            BEGIN
              NO:=((N MOD 4) DIV 3);
            END;
        END;
{1|4} IF (F[J]=4) THEN
        BEGIN
          IF (S[I]=0) THEN
            BEGIN
              NO:=((N MOD 4) DIV 3);
            END
          ELSE
            BEGIN
              NO:=((N MOD 8) DIV 7);
            END;
        END;
      if (NO<>0) then WRITE(fx,NO:1,'|')
                 else WRITE(fx,' |');
    END;
    WRITELN(fx);
 END;
 WRITELN(fx,'└──────┴──┴──┴──┴──┴──┴──┴──┴──┘');
END;
CLOSE(fx);
END.
```



Ligne de métro avec deux fourches

F=1|1	STATIONS							
TEMPS	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2			1	1	1	1		
3	1	1	1	1	1	1	1	1
4			1	1	1	1		
5	1	1	1	1	1	1	1	1
6			1	1	1	1		
7	1	1	1	1	1	1	1	1
8			1	1	1	1		
9	1	1	1	1	1	1	1	1
10			1	1	1	1		
11	1	1	1	1	1	1	1	1
12			1	1	1	1		
13	1	1	1	1	1	1	1	1
14			1	1	1	1		
15	1	1	1	1	1	1	1	1
16			1	1	1	1		
17	1	1	1	1	1	1	1	1
18			1	1	1	1		
19	1	1	1	1	1	1	1	1
20			1	1	1	1		

F=1|2	STATIONS							
TEMPS	1	2	3	4	5	6	7	8
1			1	1	1	1		
2								
3	1	1	1	1	1	1	1	1
4								
5			1	1	1	1		
6								
7	1	1	1	1	1	1	1	1
8								
9			1	1	1	1		
10								
11	1	1	1	1	1	1	1	1
12								
13			1	1	1	1		
14								
15	1	1	1	1	1	1	1	1
16								
17			1	1	1	1		
18								
19	1	1	1	1	1	1	1	1
20								

F=1|4	STATIONS							
TEMPS	1	2	3	4	5	6	7	8
1								
2								
3			1	1	1	1		
4								
5								
6								
7	1	1	1	1	1	1	1	1
8								
9								
10								
11			1	1	1	1		
12								
13								
14								
15	1	1	1	1	1	1	1	1
16								
17								
18								
19			1	1	1	1		
20								

Résultats de simulation

ANNEXE 1.2 - PRÉSENTATION DU MODULE RESOLV[70]

Nous présentons ici le principe général du module RESOLV qui n'est pas encore totalement programmé.

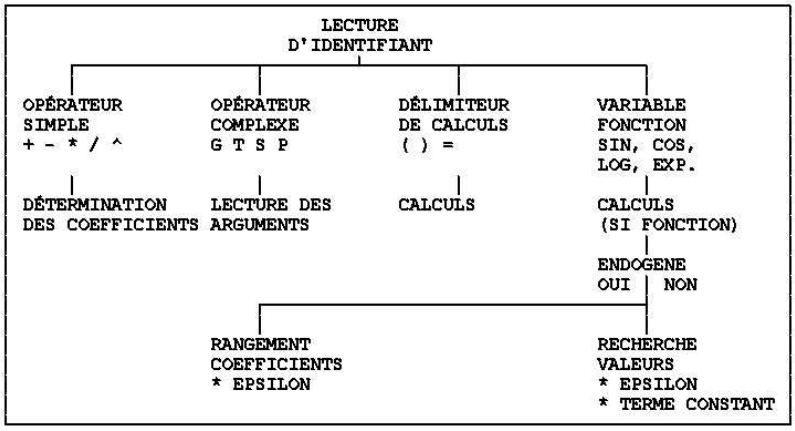


Fig.1.2.b - Organigramme de traitement des équations comptables et de définition

Le traitement d'une ligne d'instruction (une équation) se déroule différemment selon que l'équation est une équation comptable ou de définition d'une part, économétrique d'autre part - voir Fig.1.2.b et c.

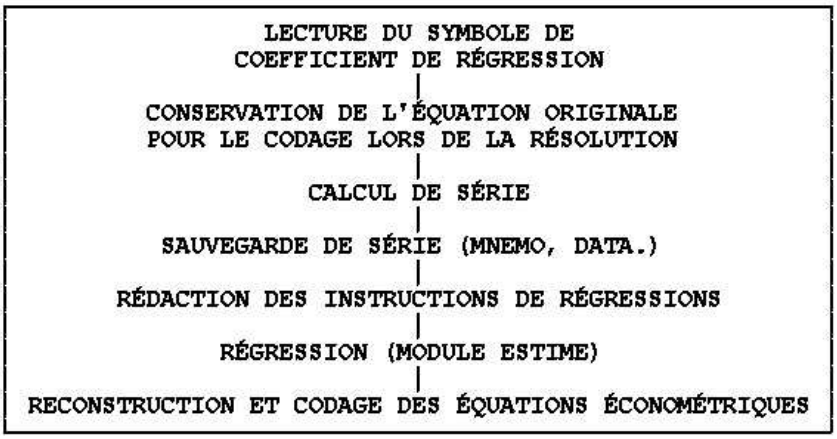


Fig.1.2.c - Organigramme de traitement des équations économétriques

Dans la pratique le travail de résolution ne sera pas nécessairement aussi "linéaire", dans la mesure où certains coefficients apparaissent plusieurs fois dans les équations. Le système doit donc être remanié s'il l'on ne veut pas effectuer plusieurs traitements. Les équations doivent être codées selon une syntaxe précise - TABLEAU N°1.2.a. L'exemple qui suit, Modèle Klein-Goldberger, permet de mettre en évidence la syntaxe.

[70]- Version provisoire d'après notre note de travail (1993.c).

TABLEAU N°1.2.a - Codage des équations avec RESOLV

SYNTAXE	SIGNIFICATION
VBL	VARIABLE VBL
VBL[i,j,k]	ELEMENT i,j,k DE LA VARIABLE VBL
θ[VBL]	TAUX DE CROISSANCE DE VBL
α	COEFFICIENT DE REGRESSION
Γ[p,VLB]	DECALAGE p DE VBL
δ[p,VBL]	DIFFERENCE pIEME DE VBL
$+ - /* \hat{}$	OPERATEURS ARITHMETIQUES
( )	DELIMITEURS D'OPERATIONS
[ ]	DELIMITEURS D'ARGUMENTS
{ }	DELIMITEURS DE SOMMATION / PRODUIT
Σ {b1,b2,VBL[k]}	SOMME DE VBL[k] POUR k=b1 A k=b2
Π {b1,b2,VBL[k]}	PRODUIT DE VBL[k] POUR k=b1 A k=b2
SINUS	FONCTION SINUS
COSINUS	FONCTION COSINUS
LOGAR	FONCTION LOGARITHME NEPERIEN
EXPON	FONCTION EXPONENTIELLE

La première colonne est réservée à l'identification des équations[71]. L'énoncé des équations commence colonne 11. Le symbole $ en fin de ligne permet de continuer à la ligne suivante (colonne 11). Chaque équation est lue comme une séquence d'identifiant - opérateurs simples ou complexes, délimiteurs, variables, fonctions ou coefficients de régression.

Fig.1.2.a - Codage des équations du modèle Klein-Goldberger avec RESOLV

```
*******************************************************************************
********************* MODELE DE KLEIN-GOLDBERGER SIMPLIFIE *********************
********* P.ARTUS ET AL. (MODÉLISATION MACROÉCONOMIQUE, 1986 PP.14-21) *********
*******************************************************************************

A  1 : CC + I + G = Y + T + D
A  2 : W1 + W2 + P + A1 + A2 = Y
A  3 : δ[1,K] = I - D
A  4 : δ[1,B] = SP
A  5 : HW * ( W / P ) * NW = W1 + W2
B  6 : CC = α * Γ[1,CC] + α * ( W1 + W2 - TW ) $
        + α * ( P - TP - SP ) + α * ( A1 +A2 - TA ) $
        + α * Γ[1,L] + α * NP + CONST
B  7 : I = α * Γ[1,( P - TP + A1 + A2 - TA + D )] $
        + α * Γ[1,K] + α * Γ[1,L2] + CONST
B  8 : W1 = α * ( Y + T + D - W2 ) $
        + α * Γ[1,( Y + T + D - W2 )] + α * TREND + CONST
B  9 : D = α * ( K + Γ[1,K] ) α * ( Y + T + D - W2 ) + CONST
B 10 : Y + T + D - W2 = α * ( HW * ( NW - NG + NE )) $
        + α * ( K + Γ[1,K] ) + α * TREND + CONST
B 11 : PC = α * P + CONST
B 12 : SP = α * ( PC - TC ) + α * Γ[1,B] + CONST
B 13 : L1 = α * ( W1 + W2 - TW + P - TP - SP + A1 + A2 - TA ) + CONST
B 14 : L2 = α * Γ[1,L2] + α * W1 - α * ( PX - Γ[1,PX] ) + CONST
B 15 : W - Γ[1,W] = α * ( N - NW - NE ) $
        + α * ( Γ[1,PX] - Γ[2,PX] ) + α * TREND + CONST
B 16 : A1 * ( P / ( PA )) $
        = α * ( W1 +W2 - TW + P - TP - SP) * ( P / ( PA )) $
        + α * FA
B 17 : PA = CONST + α * PX
```

[71]- C=équation comptable, D= équation de définition, E=équation économétrique, $=suite d'une équation, tout signe sauf A, B, D, ou " "= commentaire.

II/ Les Algorithmes d'Organisation, de Gestion et d'Analyse des Données

Les données économiques tant quantitatives que qualitatives constituent des informations relatives à des réalités économiques. La modélisation consiste d'une part, très concrètement, à structurer ces informations pour les rendre les modèles opérationnels, mais également à tirer le maximum de ces données *i.e.* les informations pertinentes. C'est pourquoi, la modélisation recourt et à des algorithmes d'organisation des données - *i.e.* de tri, de classement en vue de leur meilleure accessibilité, et à des algorithmes d'analyse statistique de données - *i.e.* procédure de synthèse de l'information pertinente. En outre, le modélisateur doit également exploiter les techniques de représentation graphique des ces données et résultats de simulation.

a) LES ALGORITHMES D'ORGANISATION DES DONNÉES

Les données sont la source, sinon la raison d'être des modèles empiriques (tant macro que micro). On comprend donc que la gestion de leur organisation est capitale pour garantir la qualité des résultats des modèles qui les utilisent. Il s'agit de trier, de tranformer les données existantes, mais la pratique de collecte des statistiques montre que le modélisateur ne dispose pas toujours de l'intégralité de données sur lesquelles il auraît aimer pouvoir compter. Certains algorithmes permettent de combler les défaillances, voir de reconstituer une banque de données à partir des données existantes[1]. Nous aborderons enfin le problème des grands tableaux que les capacités techniques des ordinateurs ne permettent pas de traiter en une seule fois.

i - Les algorithmes de tri et de recherche

On entend par algorithme de tri, des algorithmes permettant de trouver une occurrence déterminée dans une banque ou un fichier de données, ainsi que des algorithmes permettant à partir d'un critère déterminé, de présenter (afficher) ou d'organiser (ranger) des données selon l'ordre correspondant au critère choisi. Les algorithmes que nous allons présenter ici sont utilisés tels quels ou adaptés dans notre module de gestion de banque de données GEBANK (note de travail, 1995.a).

Algorithmes de tri

Le tri consiste à ordonner les éléments quantitatifs (ou quantifiables) d'un tableau, selon un ordre croissant ou décroissant. Les tris par extraction ou par insertion consistent à déplacer un élément pour une place plus pertinente (N.WIRTH - inventeur du langage Pascal - 1987, pp.71-99). Les tris par sélection (permutation) fonctionnent sur le principe de l'échange systématique de deux éléments judicieusement choisis : le tri à bulles est l'algorithme classique de cette méthode - voir Fig.13. Ainsi par exemple, dans l'ordre croissant, les éléments les plus grands sont déplacés case après case à la fin du tableau. On affine la méthode - Méthode de Sélection simple - en amorçant le tableau par

[1]- A propos des tâches d'organisation et de gestion des données d'une banque de modèle économétrique, voir J.L.BRILLET (1994, *op.cit.*, pp.37-45)

la valeur minimale ou maximale (resp.) du tableau dans l'ordre croissant ou décroissant (resp.).

```
I :=0 ;
WHILE I<N DO
  BEGIN
    FOR J :=N DOWNTO I+1 DO
      BEGIN
        IF (T[J]<T[J-1]) THEN
          BEGIN
            VTEMP :=T[J] ;
            T[J] :=T[J-1] ;
            T[J-1] :=VTEMP ;
          END ;
        I :=I+1 ;
      END ;
  END ;
```

Fig.13 - Algorithme Pascal du tri à bulles

S'il existe une structure hiérarchique au sein du tableau - *i.e.* les éléments du tableau peuvent être représentés par un arbre - on peut recourir à des algorithmes spécifiques de tri (D.BEAUQUIER et al., 1992, pp.121-44).

TABLEAU N°2- Comparaison des méthodes de tri simple[2]

	MIN	MOY	MAX
INSERTION SIMPLE	$C = n - 1$ $M = 2(n-1)$	$C = \frac{n^2+n-2}{4}$ $M = \frac{n^2-9n-10}{4}$	$C = \frac{n^2-n}{2} - 1$ $M = \frac{n^2-3n-4}{2}$
EXTRACTION SIMPLE	$C = \frac{n^2-1}{2}$ $M = 3(n-1)$	$C = \frac{n^2-n}{2}$ $M = n.(Ln(n) + 0.57)$	$C = \frac{n^2-n}{2}$ $M = \frac{n^2}{4} + 3(n-1)$
PERMUTATION SIMPLE	$C = \frac{n^2-n}{2}$ $M = 0$	$C = \frac{n^2-n}{2}$ $M = (n^2-n) * 0.75$	$C = \frac{n^2-n}{2}$ $M = (n^2-n) * 1.5$

[2]- D'après N.WIRTH (*op.cit.*, p.97). C et M (resp.) représentent les clés de comparaisons et les mouvements nécessaires pendant le tri, tandis que n est la taille du tableau à trier.

Recherche de données

En présence d'un tableau non trié, la méthode la plus intuitive est la méthode dite "séquentielle". Elle consiste tout simplement à consulter successivement chaque élément du tableau pour tester s'il s'agit de l'élément recherché ou non[3]. La seule difficulté que présente cette technique est celle des éléments présents en multiples exemplaires - voir Fig.14, l'algorithme à gauche propose un arrêt à la première occurrence ($X = X_0 \Rightarrow$ fin de la boucle), tandis que l'algorithme à droite ne s'arrête qu'à la fin du tableau ($i \geq i_{max}$) ; les deux algorithmes indiquent le(s) rang(s) i correspondant(s). Notre programme GEBANK fonctionne sur le principe séquentiel. L'inconvénient est la lenteur d'accès, surtout si l'occurrence cherchée est à la fin du tableau, on a donc toujours intérêt à travailler sur des petits tableaux purgés des données inutiles, lorsque cela est possible (*Ibid.*, pp.337-77).

```
I :=0 ;                              I :=0 ;
REPEAT                               REPEAT
  I :=I+1 ;                            I :=I+1 ;
  IF (X=X0) THEN WRITELN(I) ;          IF (X=X0) THEN WRITELN(I) ;
UNTIL (X=X0) ;                       UNTIL (I>=IMAX) ;
```

Fig.14 - Algorithmes de recherche séquentielle

Si le tableau est déjà trié, et que les enregistrements y sont stockés selon un ordre logique (croissant, décroissant etc.), la recherche peut être plus rapide. On peut commencer par la fin du tableau si l'on a de bonnes raisons de penser que l'occurrence se trouve en fin de tableau. La technique la plus pertinente est la technique "dichotomique". On scinde le tableau en deux sous-tableaux d'égale taille et l'on teste celui de droite (ou celui de gauche) - il s'agit d'une technique analogue à celle de recherche dichotomique de $f(x) = 0$. Des techniques convergeant plus rapidement existent, cependant elles sont en même temps plus risquées, en particulier s'il n'y a pas une relation d'ordre strict dans le tableau. Il est parfois possible d'établir une fonction entre les éléments du tableau et leur position (par ex. : si les éléments sont des mots, une fonction du nombre de lettres, du rang de la première lettre dans l'ordre alphabétique etc.) - c'est la technique du hachage. L'accès à un élément est directement lié à un calcul. Cette technique reste cependant limitée dans ces applications. Dès que l'on souhaite agrandir le tableau, le risque de collision d'adresse surgit (il n'y a plus de bijection entre les éléments et les adresses). D'autre part, l'emploi de fonctions de hachage trop sophistiquées aboutit à retomber dans le problème initial, à savoir un accès séquentiel aux éléments du tableau (T.CORMEN et al., 1994, pp.216-38 ; P.HERNERT, *op.cit.*, 121-25).

ii - Les algorithmes de transformation de données

Par transformation des données, nous entendons l'application d'une fonction qui modifie intégralement une série statistique. Il peut s'agir de passer d'une série

[3]- Le module EXTRAC, qui a pour fonction de préparer les équations économétriques à estimer, effectue de nombreuses recherches de données sur différents critères (de variable pertinente, temporelles, régionaux, etc.).

en niveau vers une série en taux de croissance, ou bien encore d'effectuer une transformation comptable. Plus élaborée est l'agrégation des données.

La transformation simple des données

Il n'y a pas, à proprement parler d'algorithme de transformation de données, si ce n'est la fonction que l'on se propose d'appliquer de manière homothétique sur les données (ex. : le passage d'une série en niveau X vers une série en taux de croissance $\dot{X}$, avec suppression d'une observation). Le problème algorithmique qui se pose consiste à savoir comment programmer cette transformation de manière à ce que les calculs soient entièrement automatisés. L'une des méthodes employée par les modélisateur qui programment eux-mêmes leurs modèles de simulation, est la technique de la génération de code[4]. A titre d'illustration, nous prendrons l'exemple de notre module PROGEN. L'algorithme procède de la manière suivante : le module PROGEN lit - de gauche à droite - une ligne instruction de transformation - *i.e.* la formule de calcul à appliquer à la série. Cette formule est alors "décomposée" en deux types d'éléments : les variables et les opérateurs, ce qui permet la traduction de cette ligne d'instruction en langage Turbo-Pascal sous forme d'un un module transitoire (COMBIN) ; ce dernier est ensuite compilé puis exécuté ce qui a pour conséquence de construire la série transformée. Ainsi, l'instruction dans la syntaxe de notre système :

$$\boxed{\texttt{t NEWV1} \;=\; (\texttt{OLDV1} * \texttt{OLDV2})/(\texttt{OLDV3} - \texttt{OLDV4});}$$

devient en langage Pascal dans notre module COMBIN.PAS calculé ad hoc par PROGEN[5] :

```
FOR i :=1 TO siz DO
BEGIN
  NEWV1[i] := (OLDV1[i]*OLDV2[i])/(OLDV3[i]-OLDV4[i]) ;
END ;
```

L'agrégation des données

Dans certaines circonstances, le modélisateur souhaite diminuer la taille de son échantillon de données en recourant à une opération d'agrégation. En particulier, une variable typique de modélisation régionale est la matrice de migrations inter-régionales inter-censitaires. Il s'agit d'une matrice carrée M composée d'éléments $m^t_{r,r'}$ - effectifs de ménages ayant migré de la région r vers la région r' au cours de la période t. Le découpage administratif est parfois trop grand pour être retenu par le modélisateur. Il faut donc trouver une matrice de passage qui permette d'agréger M de dimension $(h,h)/M = (m^t_{r,r'})$ en une matrice N de dimension $(k,k)/N = (n^t_{s,s'})$ où s et s' sont les régions du nouveau découpage. Ce calcul revient à un problème matriciel de type permutations. En effet, soit la matrice rectangulaire P de dimension $(k,h)/P = (p_{i,j})$ où $p_{i,j}$ est égal à 1

[4]- Voir nos notes (1994.b & d). Pour une présentation rigoureuse des langages utilisateurs, voir I.DANAILA et al. (2003, pp.285-315).

[5]- Voir en Annexe 2.1 la traduction complète de la l'instruction lorsque les variables comportent trois dimensions.

si la région r du découpage de M appartient à la région i du découpage de N, et à 0 sinon. Il vient alors facilement que $N = P.M.{}^tP$, avec ${}^tP$ est la matrice transposée de P. Bien que mathématiquement élégante, cette solution n'en est pas moins trop lente[6] et trop coûteuse en termes de place mémoire (2 matrices). C'est pourquoi nous avons proposé l'algorithme d'Agrégation naïve suivant :

```
VC1 :=VC[1] ;
FOR n :=1 TO h DO BEGIN
  FOR i :=VC1 TO VC[n] DO BEGIN
    VL1 :=VL[1] ;
    FOR m :=1 TO h DO BEGIN
      FOR j :=VL1 TO VL[m] DO BEGIN
        N[i,j] :=N[i,j]+M[n,m] ;
      END ;
      VL1 :=VL[m+1] ;
    END ;
    VC1 :=VC[n+1] ;
  END ;
END ;
```

Fig.15 - Procédure d'Agrégation naïve

à tout triplet (M, V^l, V^c) associe N telle que

$$n_{i,j} = \sum_{u \in C_{i,j}} \sum_{v \in C_{i,j}} m_{u,v}$$

par itérations successives sur les dimensions de M. Les vecteurs V^l et V^c (resp.) contiennent les clés de passage en lignes, en colonnes (resp.) de M vers N. Leurs composantes v_i^l , v_i^c (resp.) sont telles que $v_i^l = j$ si la i-ème région de M appartient à la j-ème région de N (Pour plus d'informations voir Annexe 2.2. ainsi que nos notes 1993.a, 1995.b et 1996.a).

iii - Les algorithmes de reconstitution de données

Les techniques de reconstitution de données manquantes consistent à essayer de s'approcher le plus possible de la logique d'organisation des données d'un échantillon pour déterminer au plus juste la valeur d'une donnée manquante (ou inobservable). Nous présenterons les techniques classiques d'interpolation, puis nous aborderons des techniques spécifiquement utilisées en statistiques économiques et sociales, à savoir les méthodes RAS et ASAM. La reconstitution de données manquantes pose deux problèmes. Le premier est naturellement celui de la carence d'information qu'il faut traiter. Le second est qu'il faut que l'information reconstituée soient bien intégrée dans son échantillon de données. En d'autres termes, à la logique d'organisation des données originelles, il ne faut pas substituer une logique inadéquate. Un exemple classique de hiatus est celui d'une série chronologique dont les valeurs croissent selon une fonction quadratique et sur laquelle on pratique une interpolation linéaire.

Données manquantes et interpolation

[6]- La méthode d'agrégation naïve d'une matrice 30x30 dure 11' contre 3'40 avec la méthode matricielle (test effectué sur un ordinateur 16 MHz - voir note 1993.a).

L'interpolation consiste à intercaler entre deux données successives une nouvelle donnée censée remplacer la donnée manquante. Un certain nombres d'hypothèses doivent être faites sur l'évolution probable des données pour proposer la détermination de la donnée manquante. L'interpolation linéaire est la plus simple d'entre elles. Elle est basée sur le fait que l'on suppose que les coordonnées de la donnée manquante (x_i, y_i) se situent sur la droite formée par les points de coordonnées (x_0, y_0) et (x_1, y_1).

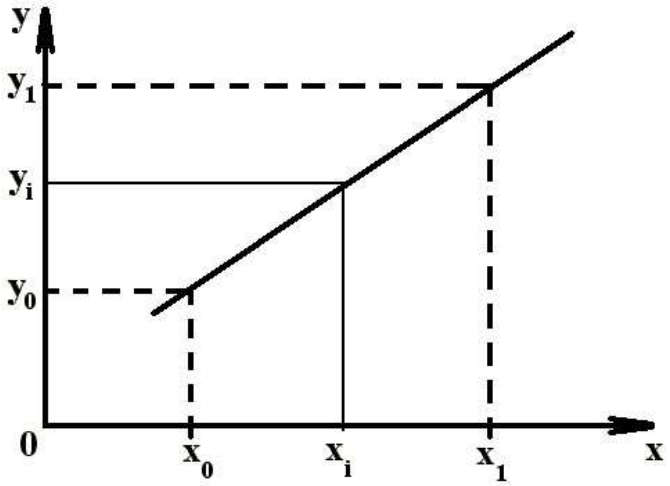


Fig.16 - Interpolation linéaire

Plus généralement, on peut effectuer des interpolations basées sur des hypothèses d'évolution polynômiale (J.G.DION et R.GAUDET, *op.cit.*, pp.163-250 ; R.DONY *op.cit.*, pp.171-92). Si l'on dispose de n couples (x_i, y_i) avec $i \in [1, n]$, l'interpolation est une opération[7] permettant d'obtenir un polynôme[8]. On détermine les coefficients du polynôme en résolvant le système linéaire $V.a = Y$ suivant :

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} Y_0 \\ \vdots \\ Y_n \end{pmatrix}$$

où V est la matrice de Vandermonde. Ce type de résolution est assez fastidieux, et impose une nécessaire égalité entre le nombre de points et le degré du polynôme . C'est pourquoi d'autres techniques on été proposées telle que celle de Lagrange :

$$P(x) = \sum_{i=0}^{n} y_i.L_i(x)$$

[7]- La technique de lissage n'impose pas l'égalité, mais la minimisation des écarts - les techniques économétriques relèvent du lissage (Cf. Infra). En prévision, le lissage consiste à utiliser une valeur observée et une valeur prédite pondérée pour déterminer une nouvelle valeur (voir la généralisation à n observations de R.BROWN, 1962).

[8]- En cas de sous-dimensionnement, on peut imposer des contraintes supplémentaires, notamment sur les dérivées.

avec

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right)$$

- voir programmes en Annexe 2.1. Nous n'exposerons pas ici l'ensemble des techniques d'interpolation ; signalons seulement qu'en dehors du problème de dimensionnement du polynôme, se pose le problème de la pertinence de l'ajustement global[9]. En d'autres termes, il vaut parfois mieux faire une "interpolation polynômiale par morceaux" - cette technique s'appelle la technique des splines. Au lieu de chercher un polynôme on en cherche plusieurs sur des intervalles exclusifs et l'on introduit des conditions de raccord (R.THÉODOR, *op.cit.*, pp.131-38). On obtient un algorithme[10] basé sur les équations suivantes :

$$P(t) = \sum_{i=0}^{n} P_i . N_{i,j}(t)$$

$$N_{i,j}(t) = \frac{(t - x_i).N_{i,j-1}(t)}{x_{i+j-1}} + \frac{(x_{i+j} - t).N_{i+1,j-1}(t)}{x_{i+j} - x_{i+1}}$$

$$N_{i,j}(t) = \begin{cases} 1 \; si \; x_t \leq t \leq x_{t+1} \\ 0 \; sinon \end{cases}$$

Données manquantes, équilibrage de tableaux avec marges connues

Le problème du traitement de données statistiques peut consister à combler des lacunes, ou bien à équilibrer le tableau selon des critères jugés objectifs. Certaines méthodes relèvent des techniques économétriques, d'autres sont moins élaborées et consistent seulement à caler le chiffrage des tableaux sur une sorte d'équilibre comptable. Nous n'évoquerons ici que cette seconde catégorie[11].

Données manquantes - Le traitement des lacunes se présente, dans le cas le plus simple de la manière suivante : on connaît la somme en ligne ou la somme en colonne - *i.e.* les marges. Si l'on note M_i^l et M_j^c (resp.) les marges à la ligne i et à la colonne j (resp.), x les données connues et $\bar{x}$ les données inconnues[12], on a :

$$M_i^l = \sum_{p=1}^{P_i} \bar{x}_p + \sum_{r=1}^{R_i} x_r$$

et

$$M_j^c = \sum_{q=1}^{Q_j} \bar{x}_q + \sum_{s=1}^{S_j} x_s$$

[9]- Une technique couramment employée consiste à ajuster la série dont une donnée est manquante, sur une autre série réputée évoluer de manière analogue.

[10]- Tiré de R.DONY (1986, pp.142-70). On y trouvera des programmes en langage Basic.

[11]- A propos des techniques statistiques de traitement des données manquantes, voir R.J.A.LITTLE et D.B.RUBIN, Statistical Analysis with Missing Data, New York, J.Wiley, 1987, 278 p.

[12]- Pour simplifier la présentation, nous raisonnons ici avec un tableau carré. Avec un tableau rectangulaire, nous raisonnerions avec le rang de la matrice.

$\forall i \in [1, N]$ et $\forall j \in [1, M]$. On voit que le système n'est soluble que si le nombre d'inconnues ne dépasse pas le nombre d'équations N, c'est-à-dire si

$$\sum_{i=1}^{N} P_i = \sum_{j=1}^{M} Q_j \leq N$$

Equilibrage - Dans la pratique des statistiques, on dispose de données que l'on va chercher à rendre compatibles non seulement avec ses marges observables $[M_t^l, M_t^c]$, mais avec des marges de références antérieures, notées $[M_{t-1}^l, M_{t-1}^c]$. L'algorithme RAS (R.STONE, 1970, pp.45-56) consiste, dans une première étape, à multiplier chaque ligne du tableau initial par un même coefficient, de manière à obtenir les marges voulues[13]. Dans une seconde étape, on multiplie chaque colonne par un même coefficient et on réitère la procédure jusqu'à obtenir la convergence vers ces marges de référence.

iv - Le calcul matriciel sur grands tableaux

En matière d'algorithme de calcul matriciel, le principal souci des mathématiciens consiste à trouver des méthodes de calculs plus rapides. Or, lorsque l'on souhaite programmer des calculs matriciels, le problème le plus crucial est bien souvent davantage celui de place mémoire prise par la taille des tableaux - en particulier en modélisation multidimensionnelle - que celui de la vitesse des calculs.

NOMBRES ENTIERS			
Type	Intervalle		Octets
Shortint	-128..127		8
Integer	-32768..32767		16
Longint	-2147483648..2147483647		32
Byte	0..255		8
Word	0..65535		16

NOMBRES RÉELS			
Type	Intervalle	Mantisse	Octets
real	2.9E-39..1.7E+38	11-12	6
single	1.5E-45..3.4E+38	7-8	4
double	5.0E-324..1.7E+308	15-16	8
extended	3.4E-4932..1.1E+4932	19-20	10
comp	-9.2E+18..9.2E+18	19-20	8

Fig.17 - Correspondances entre type de variable et taille mémoire en Turbo-Pascal

[13]- Voir notamment à ce propos R.COURBIS et C.POMMIER, *Construction d'un tableau d'échanges inter-industriels et inter-régionaux de l'économie française*, Paris, Economica, Coll. Modèles et macroéconomie appliquée, 1979, pp.32-35. L'algorithme RAS correspond à un programme de minimisation sous contraintes de marges - R.FROMENT, "Optimisation d'un tableau rectangulaire dont les marges sont connues", *Annales de l'INSEE*, N°9, janv.-avril, 1972, pp.49-64. Voir programmes FORTRAN dans L.J.SLATER (1972, pp.47-58).

Les systèmes informatiques (logiciels et matériels) ont progressé continûment en matière de vitesse de processeur, alors que la place mémoire disponible pour la déclaration des variables de programmes (on parle de DATA MEMORY) restait toujours verrouillée à 64 Ko. Sans prétendre à l'exhaustivité, nous allons présenter les trois techniques possibles de traitement de grandes matrices pour effectuer des calculs matriciels (factorisation pour la première et produit de matrices pour les autres). Encore faut-il nuancer ici notre propos, tous les tableaux de données, à dimensions égales, ne sont pas équivalents en termes de DATA MEMORY[14].

La technique des "matrices creuses"

On appelle "matrices creuses", des matrices présentant un très grand nombre de zéros (environ 90 à 99%). La technique des "matrices creuses"[15] est une technique qui permet de substituer des matrices de très grande taille (environ 10000x10000) par des vecteurs de plus petite taille, en posant pour principe que seuls les éléments non nuls de la matrice sont codés (P.LASCAUX, R.THÉODOR, *op.cit.*, tome 1, pp.289-327). Il existe plusieurs techniques de rangement des données[16].

1 - Le rangement aléatoire consiste à balayer une matrice creuse et à stocker les informations relatives à chaque élément non nul dans trois vecteurs, de la manière suivante :

$$A = \left(a_{i,j}\right)_{\substack{i\in[1,n]\\ j\in[1,n]}} \Rightarrow \left\{\begin{array}{l} V_1 = (a^k_{i,j})_{k\in[1,p]} \\ avec\ a^k_{i,j} \neq 0 \\ v_2 = (I_k)_{k\in[1,p]} \\ v_3 = (J_k)_{k\in[1,p]} \end{array}\right.$$

où $A(n,n)$ est une matrice de rang n, $a_{i,j}$ l'élément de la i-ème ligne j-ème colonne de A et p le nombre d'éléments non nuls de A. Le vecteur V_1 contient les p éléments non nuls de A, tandis que V_2 et V_3 (resp.) renvoient aux indices lignes et colonnes correspondants (resp.), dans la matrice A. On stocke $3p$ données au lieu des n^2 données totales.

2 - Le rangement trivial consiste, dans le cas particulier des matrices creuses symétriques, à stocker les éléments de la matrice dans l'ordre de balayage descendant.

$$A = \left(\begin{array}{cccc} 1e & & & \\ 2e & 3e & & \\ 4e & 5e & & \\ 7e & 8e & 9e & 10e \end{array}\right)$$

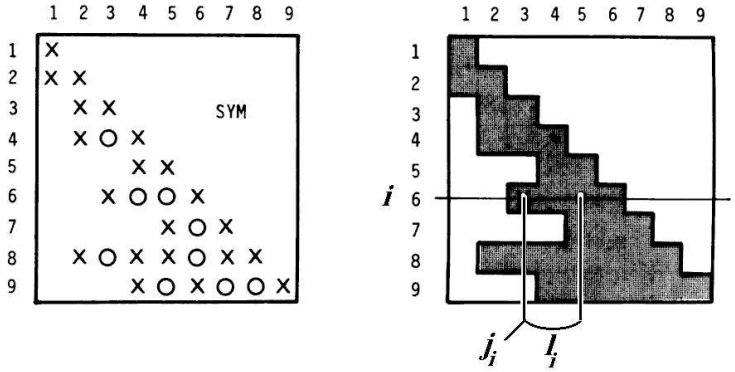
[14]- En effet, la déclaration d'un tableau d'entiers ne nécessite pas la même DATA MEMORY que celle d'un tableau de réels à dimensions égales. De même, le stockage d'un tableau de réels en simple précision nécessite moins de DATA MEMORY que celui d'un tableau en double précision. Voir le tableau des correspondances entre type de variables et place mémoire en octets (d'après BORLAND, *Turbo-Pascal 3.0 - Manuel de références*, 1987).

[15]- En modélisation macroéconométrique, on peut formuler des systèmes comportant un nombre très élevé de variables, mais en même temps avec un nombre très restreint de relations d'interdépendance.

[16]- Voir les programmes FORTRAN de L.J.SLATER (1972, *op.cit.*, pp.97-112) appliqués à la démographie (R.STONE, "Input-Output and Demographic Accounting", Minerva Vol.IV, 1966).

Ce codage permet de stocker $\frac{n(n+1)}{2}$ éléments au lieu des n^2. Le gain de place mémoire est moins intéressant que dans le cas du rangement aléatoire.

3 - Le rangement par profil consiste dans le cas particulier des matrices creuses symétriques, à effectuer un balayage ligne par ligne. A partir de ce balayage, on stocke les j_i - *i.e.* sur la ligne i, le rang de la colonne correspondant au premier élément non nul de la matrice. On stocke les l_i, appelées "demi-largeurs de bande", avec $l_i = i - j$. On stocke ainsi tous les éléments nuls et non nuls dès l'instant où ils appartiennent au profil - voir notre figure ci-après, modifiée d'après P.LASCAUX et R.THÉODORE (*op.cit.*, tome 1, p.296). Ce type de rangement présente un intérêt essentiellement dans le cas d'une factorisation $A = LD^tL$ - par exemple celle de Cholesky[17].



Matrice A **Profil (A)**

La technique du produit par blocs

La technique du produit par blocs consiste à partitionner les matrices du produit pour effectuer des produits de sous-matrices. Cette technique permet d'économiser de la place mémoire. En effet, il n'est plus obligatoire de déclarer trois matrices $A(n,n)$, $B(n,n)$ et $C(n,n)$ soit une place mémoire de $3.n^2$ mais seulement les matrices $A_{i,k}(\frac{n}{2},\frac{n}{2})$, $B_{k,j}(\frac{n}{2},\frac{n}{2})$ et $C(n,n)$, soit une place mémoire de $2.n^2$. Mais le gain est uniquement en termes de place mémoire, dans la mesure où le nombre des calculs est rigoureusement le même. Ainsi, pour une partition en deux (soient quatre blocs), on a :

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} et\ B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$$

d'où

$$C = \begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}$$

Des variantes[18] de la technique par blocs existent qui permettent de faire une économie de calculs, mais fondamentalement, l'intérêt de ce type de méthode ré-

[17]- Voir notre programme en Annexe 2.1. Pour les différences entre la Factorisation de Crout et la Factorisation de Cholesky, voir D.DUREISSEIX (2000, *op.cit.*, pp.17-18).

[18]- Citons l'Algorithme de Strassen (1969) repris in T.CORMEN et al. (1994, pp.542-47) qui revient à faire un produit récursif par bloc - avec moins de calculs que la méthode classique

side dans une meilleure allocation de la mémoire lorsque le matériel est restreint en place mémoire.

La technique des "matrices-disque"

La technique, assez intuitive, que nous proposons ici (d'après note de travail 1994.a) et que nous avons appelée "matrices-disque" tente de pallier au problème de place mémoire tout en conservant une écriture propre des algorithmes de calculs - les algorithmes doivent rester reconnaissables. En langage Turbo-Pascal, les données stockées en variables tableaux sont notées `NOMVAR`$[rang\_dim\_1, rang\_dim\_2, \ldots]$ où `NOMVAR` est le nom de la variable et $rang\_dim\_i$ est le rang de la donnée dans la i-ème dimension du tableau. Malheureusement, comme nous l'avons déjà dit, la place mémoire disponible pour le compilateur ne permet pas toujours de déclarer la taille souhaitée lorsque celle-ci dépasse les 640 Ko[19] ; problème fréquent en modélisation multidimensionnelle.

```
FOR I :=1 TO DIM1 DO BEGIN
  FOR J :=1 TO DIM2 DO BEGIN
    X[i,j] :=0. ;
      FOR K :=1 TO DIM2 DO BEGIN
        X[i,j] :=X[i,j]+A[i,k]*B[k,j] ;
      END ;
  END ;
END ;
```

Fig.18 - Procédure classique du produit de matrice

Ainsi, lorsque l'on peut déclarer les matrices X, A et B, la programmation du produit de matrice $X = A.B$ s'écrit selon la procédure décrite en Fig.18, où le i-ème-j-ème élément de X est obtenu par sommations successives des i-ème-k-ème éléments de A (lu en mémoire vive dans le tableau A) et k-ème-j-ème élément de B (lu en mémoire vive dans le tableau B) le tout étant stocké en mémoire vive dans le tableau X). Dans le cas où la mémoire est insuffisante, nous proposons de recourir à une fonction que nous avons appelée `XX` et le même programme - voir notre programme LARGEMAT en Annexe 2.1 - s'écrit comme indiqué Fig.19. Le i-ème-j-ème élément de X est obtenu par sommations successives des i-ème-k-ème éléments de A (lu dans le tableau A, stocké dans un fichier à accès direct) et k-ème-j-ème éléments de B (lu en mémoire vive dans le tableau B, stocké dans un fichier à accès direct) le tout étant stocké dans le tableau X, en fichier à accès direct).

par blocs. On pourra également voir G.LÉVY (1994, pp.385-86) ou bien encore T.CORMEN et al. (*op.cit.*, pp.725-36) à propos de la méthode de recherche de cheminement optimal des calculs matriciels par la programmation dynamique.

[19]- Ce problème est cependant, enfin en passe de changer. Notamment, le langage DELPHI, successeur de Turbo-Pascal permet d'utiliser la mémoire vive pour déclarer jusqu'à 2 Go de variable. C'est pourquoi, il est prévu, que le système que nous avons programmé bascule prochainement en langage Delphi.

```
FUNCTION XX(VAR FIL :FILE OF REAL;
                MAXSIZ1 :INTEGER;
                X1,X2 :INTEGER) : REAL;
VAR POS :INTEGER;
    DATA :REAL;
BEGIN
  POS :=(X1-1)*MAXSIZ1+X2-1;
  SEEK(FIL,POS);
  READ(FIL,DATA);
  XX :=DATA;
END;
```

```
PROCEDURE NEW_PROMAT (VAR X :MATRIX);
BEGIN
  FOR I :=1 TO DIM1 DO BEGIN
    FOR J :=1 TO DIM2 DO BEGIN
      X :=0.;
      FOR K :=1 TO DIM2 DO BEGIN
        X :=X+XX(F1,DIM1,I,K)*XX(F2,DIM1,K,J);
      END;
    END;
  END;
END;
```

Fig.19 - Fonction matrice-disque XX[20]
et procédure modifiée du produit de matrice

Les paramètres $F1$ et $F2$ renvoient aux noms de fichiers, et les paramètres $DIM1$ et $DIM2$ (resp.) renvoient aux dimensions en ligne et colonne (resp.) des matrices - voir la fonction. Ainsi, notre méthode consiste à substituer un stockage de données sur support disque (mémoire morte) d'accès relativement lent - voir TABLEAU N°3 - mais moins limité en capacité, à un stockage de données en mémoire vive plus rapide mais dont les capacités sont vite saturées.

TABLEAU N°3 - Performance des calculs matriciels avec le procédure Largmat[21]

MATRICE	DURÉE	TAILLE
35	2"	7350
50	4"	15000
75	10"	33750
100	24"	60000
150	1'20"	135000
175	2'08"	183750
200	3'10"	240000
500	57'08"	1500000

b) LES ALGORITHMES D'ANALYSE STATISTIQUE DE DONNÉES

À l'intérieur d'une banque de données, ces dernières présentent une logique, une cohérence que les outils statistiques permettent de mettre en évidence. Ces techniques sont de deux ordres : les premières cherchent à mettre en lumière des caractéristiques statistiques sans *a priori*, c'est l'analyse de données ; les secondes consistent à tester une formalisation particulière pour connaître son degré de pertinence, c'est l'économétrie.

i - L'analyse de données

L'analyse de données est un ensemble de techniques statistiques permettant de résumer des informations contenues dans un échantillon statistique. Ces techniques procèdent par décomposition de l'échantillon en composantes, par ordre

[20]- Soient M_k une hyper-matrice H_K à K dimensions $N_K^H, N_{K-1}^H \ldots, N_1^H$ où les N_i^H sont les tailles des dimensions i et V un vecteur à une dimension de taille $N^V = \prod_{j=1}^{K} N_j^H$ alors on obtient le rang r dans V d'un l'élément m_i de coordonnées $i_1, i_2, \ldots, i_K$ en posant $r = 1 + \sum_{p=1}^{K-1}(i_p - 1)\prod_{j-1}^{p-1} N_j^H$.

[21]- Avec un processeur cadencé à 700 MHz.

décroissant de pertinence - le critère de pertinence étant fourni par la contribution de la composante à la variance totale. A la base de l'analyse des données on trouve l'analyse en composantes principales.

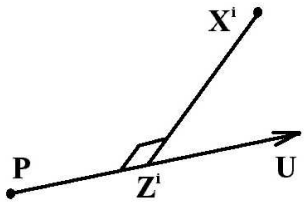
L'analyse en composantes principales

La technique de l'A.C.P. (H.HOTELLING, 1933) suppose une représentation particulière des données - *i.e.* individus/caractères. L'algorithme de l'A.C.P. procède à une transformation de l'échantillon de données de l'espace originel vers l'espace formé par les axes expliquant le mieux la variance totale ou partielle - par projection sur un nombre restreint d'axes - de l'échantillon (un nuage de points). La projection des individus sur un nombre restreint de plans a tendance à diminuer la distance entre les individus. C'est pourquoi, le critère retenu pour choisir les axes principaux a été la conservation des distances entre les individus. Les axes principaux s'obtiennent en recherchant les valeurs propres et les vecteurs propres associés au nuage de points étudié[22].

Préparation du tableau de données - A partir d'un tableau de données (individus/caractères) T, et d'une métrique M (euclidienne I_p, ou non), on calcule donc la matrice variances-covariances $V = X.D_p.{}^tX$ avec $D_p = \frac{1}{n}.I_n$, puis la matrice des corrélations $R = D_{1/\sigma}.V.D_{1/\sigma}$ avec $D_{1/\sigma}$ matrice diagonale conte-

[22]- On raisonne en effet à partir d'un "nuage de points" $N(I)$ à n points X_i dans l'espace E^k. A chaque point est associé une masse m_i, On appelle inertie de $N(I)$ par rapport au point P :

$$In_p(I) = \sum_{i=1}^{n} m_i \|X^i - P\|^2$$



On appelle inertie par rapport au point P expliquée par la direction U, l'inertie des points Z_i, projections orthogonales des X_i sur le vecteur U passant par P, si l'on associe à chaque Z_i :

$$In_p(U) = \sum_{i=1}^{n} m_i \|Z^i - P\|^2$$

La direction du premier axe factoriel est celle qui rend maximale l'expression $In(U)$, c'est-à-dire que cela revient à former le programme :

$$Max\{U'\Big(\textstyle\sum_{i=1}^{n} m_i X^i.X^{i\prime}\Big)U\}$$
$$\Big| \; U'U = 1$$

On pourrait montrer que le premier axe factoriel est le vecteur propre U_1 correspondant à λ_1, la plus petite valeur propre de V. L'inertie expliquée par cet axe est λ_1 (M.VOLLE, *Analyse des données*, Paris, Economica, Coll. Economie et statistiques avancées, pp.81-107, 1985).

nant les inverses des écarts types des caractères. On diagonalise ensuite V.

Les algorithmes de diagonalisation de matrice - L'analyse numérique propose un grand nombre d'algorithmes de calcul des vecteurs et valeurs propres (J.G.DION et R.GAUDET, *op.cit.*, pp.463-535 ; D.DUREISSEIX, *op.cit.*, pp.31-46)). Nous ne l'évoquerons pas de manière exhaustive. Nous pouvons cependant dire ici, qu'il existe deux types d'algorithmes. Une première catégorie permet de déterminer une approximation de la valeur propre de plus grand module, ou bien un nombre restreint de valeurs propres (P.LASCAUX et R.THÉODOR, tome 2, *op.cit.*, pp.591-688). Ainsi, la Méthode des puissances reprend les algorithmes de résolution itérative de systèmes d'équations linéaires . Soit A la matrice à diagonaliser, et $\{x_1, \dots x_n\}$ le système des n vecteurs propres de A, on obtient une estimation de la valeur propre dominante λ_1 associée au vecteur propre x_1 en formant l'algorithme :

$$\begin{array}{lllllll} z_1 & = & A.z_0 & & & & \\ z_1 & = & A.z_1 & = & A^2.z_0 & & \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \\ z_k & = & A.z_{k-1} & = & \dots & \dots & A^k.z_0 \end{array}$$

$\forall\ k \in [1, n]$. Le choix du vecteur initial z_0 est déterminant pour la convergence du système.

La Méthode itérative du quotient de Rayleigh consiste, après avoir posé $z_0\ /\ z_0^t z_0$, à itérer selon l'algorithme suivant (à la k-ème itération) :

$$\begin{array}{r} \lambda_1^{(k)} = z_k^t.A.z_k \\ (A - \lambda_1^k.I).y_{k+1} = z_k \\ z_{k+1} = \frac{y_{k+1}}{\|y_{k+1}\|_2} \end{array}$$

Eu égard à la rapidité des processeurs, il n'est plus aussi pertinent de se limiter à un nombre restreint de valeurs et vecteurs propres (voir P.LASCAUX et R.THÉODOR, tome 2, *op.cit.*, pp.660-688 pour d'autres méthodes). Les méthodes que nous allons sommairement présenter (Jacobi, Bissection, LU et QR) fournissent quant à elles la totalité des valeurs et vecteurs propres des matrices étudiées (P.LASCAUX et R.THÉODOR, tome 2, *op.cit.*, pp.689-754). Les itérations de la Méthode LU (H.RUSTISHAUSER, 1958) démarrent en posant $A_1 = A$, on cherche ensuite la décomposition $L.U = A_k$ telle que $L_{i,i} = 1\ \forall\ i$. Si cette décomposition est possible on réitère en formant $A_{k+1} = L_k.U_k$ sinon l'algorithme s'arrête. La Méthode QR (J.F.G.FRANCIS, 1962) initialise son algorithme en posant également $A_1 = A$, mais la décomposition retenue est $Q_k.R_k = A_k$ avec les deux conditions suivantes : $Q_k^t.Q_k = I$ et R_k est triangulaire supérieure. On réitère ensuite avec $A_{k+1} = R_k.Q_k$.

La Méthode de Jacobi procède à des transformations par rotation. On fixe p et q tels que $1 \leq p < q \leq N$ et un angle *theta* puis l'on forme la matrice $G_{p,q}(\theta)$

$$G_{p,q} = \begin{pmatrix} & & \downarrow p & & \downarrow q & & \\ 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & cos(\theta) & & sin(\theta) & & \leftarrow p \\ & & & & & & \\ & & & & & & \\ & & -cos(\theta) & & sin(\theta) & & \leftarrow q \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

l'algorithme consiste alors à itérer $A_k = G_k^t.A_{k-1}.G_k$ où k est le rang de l'itération - (M.SIBONY et J.C.MARDON, tome 1, *op.cit.*, chap.3).

Dépouillement des résultats - A partir du calcul des nouvelles composantes des individus dans les nouveaux axes $c_k =^t X.v_k$ où les v_k sont les vecteurs propres normés dans la métrique, on peut juger de la proximité des individus. Un critère de qualité de la représentation doit néanmoins être pris en compte pour éviter des erreurs d'appréciation, la contribution relative du facteur à l'inertie de l'individu : $CRT(\Delta_{u_k}, \bar{X}_j) = \frac{(c_k)_j^2}{\|\bar{X}_j\|_M^2}$. Ajoutons enfin que l'analyse de données peut également fournir des outils statistiques de traitement des données manquantes.

Les méthodes dérivées de l'A.C.P.

L'Analyse Factorielle des Correspondances est une technique qui vise à décrire les liaisons entre deux caractères qualitatifs X et Y ayant respectivement q et p modalités. C'est une analyse métrique des lignes et des colonnes d'un grand tableau de contingence croisant deux variables dont chacune définit une partition sur la population étudiée. On transforme le tableau original de données contenant des éléments $k_{i,j}$ en un tableau de $X_{i,j} = \frac{P_{i,j}}{P_{.,j}}$ où $P_{i,j} = \frac{k_{i,j}}{n.}$. Puis on effectue une A.C.P. sur le tableau X. L'Analyse des Correspondances Multiples correspond à une généralisation de l'A.F.C. dans la mesure où l'on considère cette fois plus de deux caractères (M.VOLLE, *op.cit.*, 182-205). L'Analyse Factorielle Discriminante considère des observations de p caractères quantitatifs $\{x_1, ...x_p\}$ sur une population de n individus répartis en q classes disjointes. On cherche à savoir si les p caractères permettent de différentier les q classes de chaque individu à partir de ces seuls caractères. En se ramenant à l'espace G des centres de gravités de q groupes, on obtient une A.C.P. de ce nuage de points G. Les Méthodes de Classifications cherchent, à partir des outils mathématiques présentés plus haut (inertie, distance, métrique etc.), à déterminer la partition d'un échantillon de données qui minimise l'inertie intra-classe et qui maximise l'inertie inter-classe[23]. Ajoutons que le calcul des valeurs et vecteurs propres est

[23]- Voir M.VOLLE (*op.cit.*, pp.265-315) et M.JAMBU (1978) pour les aspects théoriques,

utilisé pour analyser le comportement temporel des modèles dynamiques[24].

ii - L'économétrie

L'économétrie est comme l'analyse de données, basée sur l'algèbre linéaire et les statistiques[25]. Mais l'économétrie propose de tester sur l'échantillon de données, les formalisations les plus pertinentes - détermination des coefficients d'équations testées avec leurs encadrements statistiques. Il ne s'agit pas de présenter ici l'ensemble de la démarche économétrique, mais seulement les algorithmes qui lui sont spécifiques. Au coeur de l'économétrie se trouve les Moindres Carrés Ordinaires - *i.e.* pour un nuage de points donné, la droite qui minimise les distances entre elle-même et les points du nuage.

Moindres carrés ordinaires

A l'état "brut", la méthode des moindres carrés[26] fournit les coefficients d'une équation que l'on cherche à tester sur un nuage de points. Par ailleurs, la dimension de l'espace dans lequel on représente le nuage dépend du nombre de variables que l'on souhaite introduire dans l'équation. Algébriquement, on pose l'équation $Y = X.a$, où Y est le vecteur de la variable expliquée, a le vecteur des coefficients recherchés et X la matrice des variables explicatives, mais contrairement aux algorithmes de résolution de système, ce ne sont pas les variables que l'on cherche, mais les coefficients. La détermination du vecteur a , notée , s'obtient en formant $\hat{a} = (X'X)^{-1}.X'.Y$. On recalcule alors le vecteur de la variable expliquée noté $\hat{Y}$ par ce vecteur en formant $\hat{Y} = X.\hat{a}$. Cette méthode, que nous avons adoptée dans notre module économétrique ESTIME, présente l'inconvénient de contraindre à effectuer une inversion de matrice (Programme Pascal in D.MONASSE, *op.cit.*, pp.105-108). C'est pourquoi, certains logiciels modifient la forme de l'équation $Y = X.a$ pour obtenir une résolution d'équation en a - on parle d'équations normales -, méthode qui présente des inconvénients en termes d'arrondis. La seconde phase de la méthode économétrique est davantage statistique, en ce sens qu'il s'agit de fournir des tests statistiques permettant de juger de la qualité des coefficients trouvés. Celle-ci dépend de l'amplitude des résidus $\varepsilon = Y - \hat{Y}$. La pertinence de la formulation est fournie par le coefficient de détermination,

$$\bar{R}^2 = 1 - \frac{\frac{1}{n-k}.\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n-1}.\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

R.CÉHESSAT (ED.) (1976), L.LEBART et al. (1977), L.LEBART et al. (1982), pour des programmes FORTRAN, L.LEBART et al. (1982) pour des programmes APL, BORLAND (1987) et D.MONASSE (*op.cit.*, pp.124-27) des programmes Pascal, enfin, M.JAMBU & M.O.LEBEAUX (1978) au sujet de la classification automatique. Voir SPAD (CISIA, 1987).

[24]- P.ARTUS et al. (*op.cit., pp.123-29 et pp.154-63).*

[25]- A propos des algorithmes d'analyse numérique et de leurs liens avec l'inférence statistique, voir C.GOURIÉROUX et A. MONFORT (*Statistique et modèles économétriques*, Paris, Economica, Coll.Economie et statistiques avancées, 1989, pp.481-531).

[26]- Pour un exposé mathématique théorique et une présentation de méthodes alternatives de facrtorisation, voir P.LASCAUX et R.THÉODOR (tome 1, *op.cit.*, pp.331-84).

tandis que la pertinence variable par variable est fournie par le t-Student $t = \frac{\hat{a}}{\sigma\hat{a}}$. On obtient une première explication sur la forme des résidus, en particulier sur le fait de savoir s'ils sont corrélés (à l'ordre 1) entre eux, en utilisant la statistique de Durbin et Watson :

$$DW = \frac{\varepsilon_{t-1} - \varepsilon_{t-2}}{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Les techniques dérivées

Les autres algorithmes proposés par l'économétrie sont des combinaisons de la méthode de MCO - en particulier des procédures itératives de régression -, des techniques de simulation (la prévision consiste à effectuer le calcul de la fonction estimée sur une valeur dont la période n'a pas été observée) et des techniques de la statistique (méthodes d'ajustement de séries à des lois de probabilité). Il s'agit de se remettre dans les conditions de validité d'application du modèle des M.C.O. Du point de vue de la qualité des coefficients de régression au sens de student, on peut mettre en place une procédure itérative.

La Régression pas à pas permet de déterminer les "bons" coefficients lorsque plusieurs variables explicatives sont possibles. On classe les coefficients par ordre croissant de la valeur absolue du module et on effectue les MCO en sélectionnant toujours plus de variables parmi celles ayant les plus grands coefficients.

Algorithme de Régression pas à pas

$$MCO(Y/X_i) \rightarrow (c_i)_{i\in[1,n]}$$

$$(c_i)_{i\in[1,n]} \rightarrow \{(c_i)_{i\in[1,n]} \ / \ |c_{j-1}| \geq |c_j| \geq |c_{j+1}|\}$$

$$MCO(Y/X_k)_{p\in[1,n]} \ avec \ k \in [1,n]$$

Y, variable expliquée, X, matrice des variables explicatives et N nombre de combinaisons possibles $2n-1$, on s'arrête lorsque un des t-Student n'est plus significatif.

Concernant la qualité des résidus, on se demande s'il y a corrélation entre les résidus (à l'ordre 1, 2 ou supérieur il existe une relation entre les résidus du type

$$\varepsilon_t = \sum_{k=1}^{t-1} \rho_k . \varepsilon_{t-k}$$

A l'ordre 1, citons les méthodes de Durbin, Cochran-Orcutt et Hildreth-Lu :

Algorithme de Durbin

$$\boxed{\begin{array}{l} MCO(Y/X_i) \rightarrow \ DW \\ MCO(Y/Y_{-1}, X, X_{-1}) \rightarrow \ \hat{\rho} \\ MCO(Y - \hat{\rho}.Y_{-1}/X - \hat{\rho}.X_{-1}) \rightarrow \ (c_i)_{i\in[1,n]} \end{array}}$$

Algorithme de Cochran-Orcutt

$$\boxed{\begin{array}{l} MCO(Y/X_i) \rightarrow \ DW \\ MCO(\varepsilon_t/\varepsilon_{t-1}) \rightarrow \ \hat{\rho} \\ MCO(Y - \hat{\rho}.Y_{-1}/X - \hat{\rho}.X_{-1}) \rightarrow \ (c_i)_{i\in[1,n]} \end{array}}$$

Algorithme de Hildreth-Lu

$$\boxed{\begin{array}{l} MCO(Y/X_i) \rightarrow \ DW \\ \left\{ \begin{array}{l} \tilde{\rho} - 1 + 0.1 * k \\ MCO(Y - \tilde{\rho}_k.Y_{-1}/X - \tilde{\rho}_k.X_{-1}) \end{array} \right\}_{k\in[1,20]} \\ \rho = \tilde{\rho}_j \ / \ s_j^2 = Inf(s_k^2) \end{array}}$$

où les ρ sont les coefficients de d'auto-corrélation des résidus et les c_i sont les coefficients de régression.

Pour une régression donnée, l'abandon de l'hypothèse d'homoscédasticité des résidus ($E(\varepsilon_t) = 0$, $V(\varepsilon_t) = \sigma^2$ et $Cov(\varepsilon_t, \varepsilon_{t'}) = 0$) implique de nouvelles transformations des variables de la régression. Dans le cas de la Régression pondérée, le coefficient de pondération provient de la matrice de variance-covariance des résidus.

Algorithme de la méthode des retards polynômiaux

$$MCO(Y/X_{t-1},\ldots,X_{t-k}) \rightarrow ? \rightarrow \begin{pmatrix} \hat{a}_1 \\ \vdots \\ \hat{a}_k \end{pmatrix}$$

$$f_r/\ f_r(z) = \sum_{i=1}^{r} a_i.z^i$$

$$Y_t(a_j)_{j\in[1,n]} \rightarrow\ Y_t(\alpha_i)_{i\in[1,r]}$$

$$MCO(Y_t/\ldots) \rightarrow \begin{pmatrix} \hat{\alpha_1} \\ \vdots \\ \hat{\alpha_r} \end{pmatrix} \rightarrow \begin{pmatrix} \hat{a_1} \\ \vdots \\ \hat{a_k} \end{pmatrix}$$

Une branche de l'économétrie s'est développée autour du problème du traitement de l'autorégressivité dans le cadre de la prévision (moins que de l'explication). On peut en effet se trouver dans le cas d'une estimation du type $MCO(Y_t/X_t,\ldots,X_{t-k})$, voire $MCO(Y_t/Y_{t-1},\ldots,Y_{t-k})$. Dans le premier cas, certaines transformations - en l'occurence une interpolation polynômiale des coefficients à estimer -, telles que celle de la Méthode des retards polynômiaux de S.ALMON mais ce n'est pas la seule[27]. Ainsi, d'autres algorithmes ont été développés pour résoudre, notamment, les problèmes d'autocorrélation (sous entendu "temporelle") des erreurs à des ordres supérieurs à un. Il s'agit de tester des processus ; contrairement aux modèles, le raisonnement insiste ici davantage sur le caractère stochastique du mécanisme et l'on se focalise ainsi sur les erreurs . Les processus ainsi utilisés sont du type $MA(r)$ (*i.e.* Moving Average à l'ordre r) :

$$\boxed{Y_t = \varepsilon_t - \theta_1.\varepsilon_{t-1} - \ldots - \theta_r.\varepsilon_{t-r}}$$

ou $AR(p)$ (*i.e.* Auto-Regressif à l'ordre p) :

$$\boxed{Y_t - \phi_1.Y_{t-1} - \ldots - \phi_p.Y_{t-p} = \varepsilon_t}$$

Dans le cas de la combinaison des deux processus ; $ARMA(p,r)$:

$$\boxed{Y_t - \phi_1.Y_{t-1} - \ldots - \phi_p.Y_{t-p} = \varepsilon_t - \theta_1.\varepsilon_{t-1} - \ldots - \theta_r.\varepsilon_{t-r}}$$

l'estimation des coefficients proposée par G.E.P.BOX et G.M.JENKINS (*Time Series Analysis : Forecasting and Control*, Holden-Day, San Fransisco, 1976), est obtenue à partir du principe de maximum de vraisemblance ; toutefois, cette

[27]- Voir S.ALMON ("The Distributed Lag Between Capital Appropriation and Expenditures", *Econometrica*, jan., 1965). Si l'on souhaite tenir compte de la fréquence d'arrivée des retards on peut utiliser la méthode de L.M.KOYCK (*Distributed Lags and Investment Analysis*, Amsterdam, North-Holland, 1954). Pour un exposé des autres méthodes, voir notamment, E.MALINVAUD (*Méthodes statistiques de l'économétrie*, Paris, Dunod, Coll. Finance et économie appliquée, 1981, pp.630-56).

technique revient à celle des MCO non linéaires[28]. Citons enfin la modélisation VAR, qui revient à tester toutes les combinaisons de modèles entre une variable expliquée Y et des variables explicatives X décalées.

Pour terminer, mais non pour conclure, ajoutons que si l'immense majorité des économètres travaillent sur les problèmes d'autocorrélation temporelle des erreurs, les spécialistes d'économétrie spatiale traitent également les problèmes d'autocorrélation spatiale des erreurs[29].

Soit le modèle $Y = X.a + \varepsilon$ avec

$$y_{r,t} = x_{r,t}.a_{r,t} + \varepsilon_{r,t}$$

où r et t (resp.) sont les dimensions régionale et temporelle (resp.). Si

$$E(\varepsilon_{r,t}, \varepsilon_{r',t'}) \neq 0$$

pour

$$r \neq r'$$

alors on dit qu'il y a autocorrélation spatiale des erreurs . Il existe un test analogue à celui de Durbin, et dans la perspective du traitement de l'autocorrélation spatiale, le problème spécifique est celui de l'ordonnancement des régions. En effet, dans le cas temporel, l'ordre était naturel puisqu'il est chronologique ; dans le cas spatial on doit tenir compte de la contiguïté des régions entre elles - *i.e.* la distance des régions entre elles et l'existence ou non d'une frontière commune - Cf.Infra (§III-a-ii) à propos de la matrice de contiguïté.

RÉFÉRENCES

BEAUQUIER D., BERSTEL J., CHRÉTIENNE P., (1992), *Éléments d'algorithmique*, Paris, Masson, Coll.Manuels informatiques, 463 p.

BORLAND, (1987), *Turbo Pascal Numerical Methods Toolbox*, Borland International Inc. + Programmes.

BROWN R., (1962), *Smoothing, forecasting and Prediction*, Englewood Cliffs, Prentice-Hall.

BRILLET J.L., (1994), *Modélisation économétrique - principes et techniques*, Paris, Economica, Coll.Economie et statistiques avancées, 194 p. + Le logiciel Soritec Sampler.

CÉHESSAT R.(ED.), (1976), *Exercices commentés de statistiques et informatique appliquées*, Paris, Dunod, 460 p.

CISIA, (1987), *SPAD-N - Guide des premiers pas*, Paris, CISIA, 28 p.

CORMEN T., LEISERSON C., RIVEST R., (1994), *Introduction à l'algorithmique*, Paris, Dunod, Coll. 2ème Cycle universitaire/Ecoles d'ingénieurs, 1019 p.

DANAILA I., HECHT F., PIRONNEAU O., (2003), *Simulation numérique en C++*, Paris, Dunod, Coll. Science Sup., 340 p.

DION J.G., GAUDET R., (1996), *Méthodes d'analyse numérique - de la théorie à l'application*, Mont- Royal (Québec), Modulo, Coll. universitaire de mathématiques, 623 p.

DONY R., (1986), *Calcul des parties cachées - approximations des courbes par la méthode de Bezier et des B-Splines*, Paris, Masson, Coll. Méthodes+Programmes, 238 p.

FRANCIS J.F.G., (1962), "The QR-transformation, A Unitary Analogue to the LR-transformation", *Computer Journal*, N°4, Part.I, pp.265-71 et Part II, pp.332-45.

[28]- Voir à ce propos G.MÉLARD, "Estimation des paramètres de modèles ARMA", in J.J.DROESBEKE, B.FICHET et P.TASSI (EDS), *Séries chronologiques - Théorie et pratique des modèles ARIMA*, Paris, Economica, Coll.Association pour la statistique et ses utilisations, 1989, pp.75-91.

[29]- Pour une présentation complète, voir H.JAYET (*Analyse spatiale quantitative*, Paris, Economica, Coll. Bibliothèque de science régionale, pp.53-107).

HERNERT P., (1995), *Les algorithmes*, Paris, PUF, Coll. Que sais-je ?, 128 p.

HOTELLING H., (1933), "Analysis of a Complex of Statistical Variables into Principal Components", *Journal of Educ. Psych.*, N°24, pp.417-41 & pp.498-520.

JAMBU M., (1978), *Classification automatique pour l'analyse des données - tome 1, méthodes et algorithme*, Paris, Dunod, Coll.Décision, 310 p.

JAMBU M., LEBEAUX M.O., (1978), *Classification automatique pour l'analyse des données - tome 2, logiciels*, Paris, Dunod, Coll.Décision, 399 p.

LASCAUX P., THÉODOR R., (1986-87), *Analyse numérique matricielle appliquée à l'art de l'ingénieur (2 Vol.)*, Paris, Masson, 790 p.

LEBART L., MORINEAU A., TABART N., (1977), *Techniques de la description statistique - méthodes et logiciels pour l'analyse des grands tableaux*, Paris, Dunod, 351 p.

LEBART L., MORINEAU A., FÉNELON J.P., (1982), *Traitement des données statistiques - méthodes et programmes*, Paris, Dunod, 510 p.

LÉVY G., (1994), *Algorithmique combinatoire - méthodes constructives*, Paris, Dunod, Coll.Science informatique, 502 p. + Programmes.

MONASSE D., (1988), *Mathématiques et informatique*, Paris, Vuibert, Coll. Classes préparatoires Cours et travaux dirigés, 223 p.

RUSTISHAUSER H., (1958), "Solutions of Eigenvalue Problems with th LR-transformation", *Nat. Bur. Standards Appl. Math. Ser.*, N°49, pp.47-81.

SIBONY M., MARDON J.C., (1982), *Analyse numérique - Tome 1, systèmes linéaires et non linéaires*, Paris, Hermann, Coll.Actualité scientifique et industrielle.

SLATER L.J., (1972), *More Fortran Programs for Economists*, London, Cambridge UP, 146 p.

STONE R., (1970), *Mathematical Models of the Economy and Other Essays*, London, Chapman & Hall, 335 p.

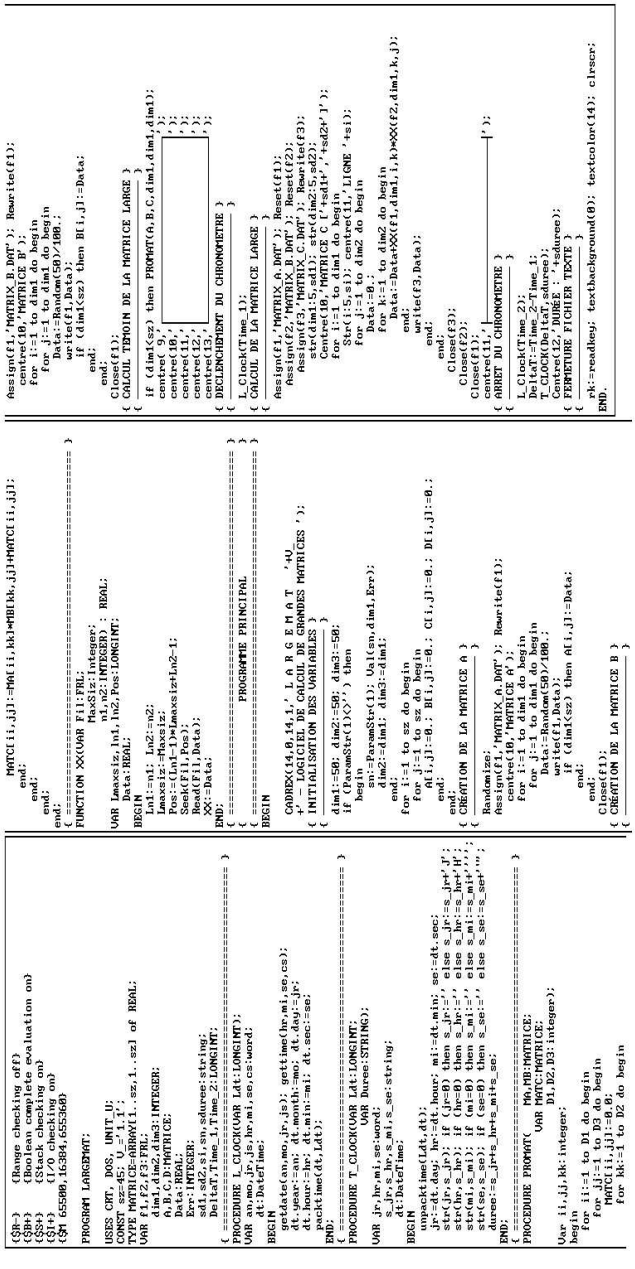
STRASSEN V., (1969), "Gaussian Elimination is not Optimal", *Numerische Mathematik*, N°14(3), pp.354-56.

THÉODOR R., (1989), *Initiation à l'analyse numérique*, Paris, Masson, Coll.CNAM Cours A, 302 p.

WIRTH N., (1987), *Algorithmes et structures de données*, Paris, Eyrolles, 320 p.

ANNEXE 2.1 - PROGRAMMES DE GESTION DE DONNÉES

Programme LARGEMAT de calcul matriciel sur grandes matrices[30]



[30]- La présentation du programme est simplifiée. Les instructions permettant le contrôle des résultats par les deux méthodes de stockage des matrices (variable de type ARRAY et matrices-disque) n'apparaissent pas ici. Les tableaux déclarés en mémoire ont servi pour effectuer les vérifications des calculs par la méthode classique.

Programme COMBIN appliqué
à l'instruction : t NEWV1 = VAR01+VAR08 ;

```
{$R-}   {Range checking off}
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}
{$M 65500,16384,655360}

PROGRAM COMBIN;
USES DOS, CRT, UNIT_U;
CONST siz1=50; siz2=50; z1=10; z2=10; z3=1;
TYPE T_MAT=ARRAY[1..siz1,1..z1,1..z2,1..z3] OF REAL;
     MAT=^T_MAT;
VAR t:ARRAY[1..3,1..siz1] OF INTEGER;
    i1,i2,i3,d1,d2,d3:INTEGER;
    VAR01,VAR08,NBMAX1:MAT;
BEGIN
{ ALLOCATION DE MEMOIRE }
{ ______________________ }
  NEW(VAR01);NEW(VAR08);NEW(NBMAX1);
{ INITIALISATION }
{ ______________ }
  CLRSCR;
  MIRO(6,11,'COMBIN 2.1 - MODULE DE COMBINAISON');
  MIRSYS(6,11);
  d1:=1; d2:=1; d3:=1;
  imax:=20;
{ LECTURE DES DONNEES }
{ ____________________ }
  ASSIGN(fx,'\SIMUL\DAT\VAR01.000');
  RESET(fx);
    i:=0;
    READLN(fx);
    READLN(fx,tf1,i1,tf2,i1,tf3,i1);
    REPEAT
      i:=i+1;
      READLN(fx);
      GOTOXY(16,10);
      WRITE('LECTURE VARIABLE [VAR011');
      FOR i1:=1 TO tf1,i1 DO BEGIN
        FOR i2:=1 TO tf2,i1 DO BEGIN
          FOR i3:=1 TO tf3,i1 DO BEGIN
            GOTOXY(16,11);
            WRITE('LECTURE ELEMENT [',i1:2,',',i2:2,',',i3:2,']');
            READLN(fx,VAR011^[i,i1,i2,i3]);
          END;
        END;
      END;
    UNTIL EOF(fx);
  CLOSE(fx);
  imax:=1;
  IF (d1<tf1,1]) THEN d1:=tf1,1];
  IF (d2<tf2,1]) THEN d2:=tf2,1];
  IF (d3<tf3,1]) THEN d3:=tf3,1];
  ASSIGN(fx,'\SIMUL\DAT\VAR08.000');
  RESET(fx);
    i:=0;
    READLN(fx);
    READLN(fx,tf1,2],tf2,2],tf3,2]);
    REPEAT
      i:=i+1;
      READLN(fx);
      GOTOXY(16,10);
      WRITE('LECTURE VARIABLE [VAR081');
      FOR i1:=1 TO tf1,2] DO BEGIN
        FOR i2:=1 TO tf2,2] DO BEGIN
          FOR i3:=1 TO tf3,2] DO BEGIN
            GOTOXY(16,11);
            WRITE('LECTURE ELEMENT [',i1:2,',',i2:2,',',i3:2,']');
            READLN(fx,VAR081^[i,i1,i2,i3]);
          END;
        END;
      END;
    UNTIL EOF(fx);
  CLOSE(fx);
  imax:=1;
  IF (d1<tf1,1]) THEN d1:=tf1,1];
  IF (d2<tf2,1]) THEN d2:=tf2,1];
  IF (d3<tf3,1]) THEN d3:=tf3,1];
  ASSIGN(fx,'\SIMUL\DAT\NBMAX1.000');
  REWRITE(fx);
  WRITELN(fx,' ');
  CLOSE(fx);
  WRITE('RESULTATS : \SIMUL\DAT\NBMAX1.000');
  FOR i:=1 to imax DO
    BEGIN
      WRITELN(fy,1960+i:4);
      FOR i1:=1 TO d1 DO BEGIN
        FOR i2:=1 TO d2 DO BEGIN
          FOR i3:=1 TO d3 DO BEGIN
            NBMAX1^[i,i1,i2,i3]:=VAR01^[i,i1,i2,i3]+VAR08^[i,i1,i2,i3];
            WRITELN(fy,NBMAX1^[i,i1,i2,i3]);
          END;
        END;
      END;
    END;
  CLOSE(fy);
{ RESTAURATION DE LA PLACE MEMOIRE }
{ _________________________________ }
  DISPOSE(VAR01);DISPOSE(VAR08);DISPOSE(NBMAX1);
END.
```

ANNEXE 2.2 - ÉTUDE DE L'ALGORITHME D'AGRÉGATION VECTORIELLE[31]

Définition 1 : *On appelle classe d'agrégation à deux dimensions $C_{i,j}$, l'ensemble des $m_{u,v}$ tels que*

$$n_{i,j} = \sum_{u \in V^l} \sum_{v \in V^c} m_{u,v}$$

où le vecteur V^l appelé vecteur d'appartenance en lignes, est composé des $v_i^l \forall\, i \in [1,h] / v_i^l = z \forall\ z \in [1,k]$ et le vecteur V^c appelé vecteur d'appartenance en colonnes, est composé des $v_j^c \forall\ j \in [1,h]\ / v_j^c = z\ \forall\ z \in [1,k]$

Définition 2 : *On appelle matrice de passage d'agrégation, la matrice P rectangle de dimension (k,h) composée des éléments $p_{i,j}$ tels que : $p_{i,j} = 1$ si la région j du découpage de M appartient à la région i du découpage de N et, $p_{i,j} = 0$ sinon.*

Définition 3 : *On appelle méthode matricielle d'agrégation l'application qui à tout couple (M,P) associe $N/N = P.M.{}^tP$,* avec ${}^tP$ est la matrice transposée de P.

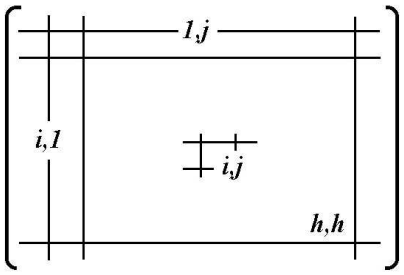


Fig.2.2.a - Balayage de M dans la cas de la méthode matricielle d'agrégation

Définition 4 : *On appelle méthode vectorielle d'agrégation l'application qui, par itérations successives sur les dimensions de M, associe N à tout triplet (M, V^l, V^c) :*

$$N = \left(n_{i,j} = \sum_{u \in C_{i,j}} \sum_{v \in C_{i,j}} m_{u,v} \right)$$

[31]- Nous aimerions remercier M. Alain TOMAZO qui a bien voulu vérifier la cohérence de ce texte (sous sa forme initiale), erreurs et/ou omissions éventuelles restant de notre entière responsabilité. De plus, M.TOMAZO a montré que l'algorithme était symétrique (ligne/colonne) et a suggéré le terme d'Algorithme d'agrégation naïve.

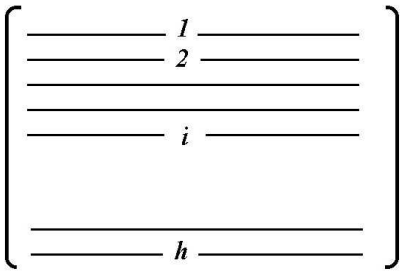


Fig.2.2.b - Balayage de M dans la cas de la méthode vectorielle d'agrégation

Définition 5 : *A toute méthode algorithmique i notée A_i est associée η_i le nombre d'itérations effectuées par l'algorithme et μ_i la taille occupée par les vecteurs et/ou matrices lors des calculs.*

Définition 6 : *On appelle vecteur d'effectifs des classes en lignes le vecteur E^l composé des $e_i^l / e_i^l = Card(C_{i,*})$, avec $e_i^l \neq 0$.*

Définition 7 : *On appelle vecteur d'effectifs des classes en colonnes le vecteur E^c composé des $e_j^c / e_j^c = Card(C_{*,j})$, avec $e_j^c \neq 0$.*

REMARQUE : Les raisonnements que nous allons présenter sont symétriques pour les lignes et les colonnes. Aussi, par souci de clarté dans les notations, nous ne reporterons pas systématiquement les indices de ligne et de colonne des éléments des vecteurs effectifs.

Hypothèse 1 : *Soit M la matrice à agréger et N la matrice résultat*

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,h} \\ m_{2,1} & m_{2,2} & & \vdots \\ \vdots & & m_{u,v} & \vdots \\ m_{h,1} & \dots & \dots & m_{h,h} \end{pmatrix} \text{ et } N = \begin{pmatrix} n_{1,1} & n_{1,2} & \dots & n_{1,k} \\ n_{2,1} & n_{2,2} & & \vdots \\ \vdots & & n_{i,j} & \vdots \\ n_{k,1} & \dots & \dots & n_{k,k} \end{pmatrix}$$

nous supposerons que $1 < k < h$.

En effet, si $k = 1$ alors

$$A_v(M) = \sum_{u=1}^{h} \sum_{v=1}^{h} m_{u,v}$$

et si $k = h$ alors $A_v(M) = M$, deux cas particuliers qui ne présentent aucun intérêt dans le cadre de nos applications économiques.

Hypothèse 2 : *Les éléments d'une même classe $C_{i,j}$ sont contigus deux à deux dans la matrice M, c'est-à-dire que*

$$\left.\begin{array}{l} m_{u,v} \in C_{i,j} \\ m_{u+2,v} \in C_{i,j} \\ m_{u,v+2} \in C_{i,j} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} m_{u+1,v} \in C_{i,j} \\ m_{u,v+1} \in Ci,j \end{array}\right.$$

Proposition 1 : *L'algorithme A_v de la méthode vectorielle est équivalent à celui Am de la méthode matricielle.*

Démonstration - Le terme général $n_{i,j}$ s'obtient en sommant les $m_{u,v}$ de M appartenant à la classe $C_{i,j}$ que l'algorithme trouve, après avoir rencontré les $m_{s,w}$ (avec $s < u$ et $w < v$) appartenant aux classes antérieures. Il vient donc :

$$n_{i,j} = \sum_{u=u_1}^{u_2} \sum_{v=v_1}^{v_2} m_{u,v} \tag{1}$$

avec

$$u_1 = \sum_{s=1}^{i-1} e_s + 1 \tag{2}$$

$$u_2 = \sum_{s=1}^{i} e_s \tag{3}$$

$$v_1 = \sum_{w=1}^{j-1} e_w + 1 \tag{4}$$

$$v_2 = \sum_{w=1}^{j} e_w \tag{5}$$

On conçoit assez facilement que dans l'équation (1), les sommations restreintes aux bornes de validité de la classe du nouveau découpage, correspondent à une sommation plus générale où les éléments n'appartenant pas à la nouvelle classe seraient multipliés par 0. C'est-à-dire que l'équation (1) est alors équivalente à

$$n_{i,j} = \sum_{u=1}^{h} \sum_{v=1}^{h} (d_{i,v}.m_{v,u}).d'_{u,j} \tag{6}$$

où les $d_{i,v}$ sont égaux à 1 si les $m_{v,u}$ appartiennent à la nouvelle classe et sont égaux à 0 sinon. On reconnaît en l'occurence la matrice de passage P composée des $p_{i,j}$
Q.E.D.

Définition 8 : *On dit que $\{A_i \succ A_j\}$ si $\eta_i < \eta_j$ et $\mu_i < \mu_j$*[32].

Définition 9 : *On dit que $\{A_i \succ A_j\}_\eta$ si $\eta_i < \eta_j$*[33].

Définition 10 : *On dit que $\{A_i \succ A_j\}_\mu$ si $\mu_i < \mu_j$*[34].

[32]- A_i strictement supérieur à A_j.
[33]- A_i partiellement supérieur à A_j au sens de η.
[34]- A_i partiellement supérieur à A_j au sens de μ.

Définition 11 : *On appelle méthode intermédiaire d'agrégation l'application, définie par les équations (1), (2), (3), (4) et (5), qui à tout triplet* (M, E^l, E^c) *associe* N *par itérations successives sur les dimensions de* N.

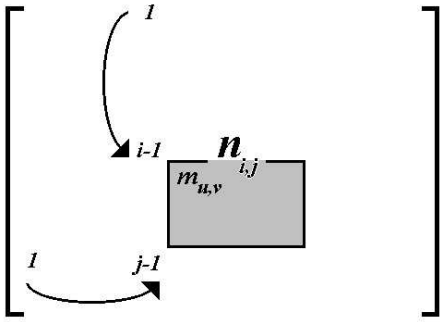


Fig.2.2.c - Balayage de M dans la cas de la méthode intermédiaire d'agrégation

Proposition 2 : *L'algorithme* A_v *de la méthode vectorielle d'agrégation est strictement supérieur à celui* A_m *de la méthode matricielle.*

Démonstration - La démonstration se déroule en deux temps.

1° - L'algorithme matriciel opère les produits $P.M$ ($k.h.h$ itérations) et $(PM).{}^tP$ ($h.k.k$ itérations). D'où $\mu_m = k.h^2 + k^2h$. Dans le cas vectoriel l'algorithme itère h^2 fois, *i.e.* une seule fois par $m_{i,j}$, affectant précisément grâce à V^l et V^c les valeurs rencontrées à la classe correspondante. D'où $\mu_v = h^2$.

2° - La place mémoire occupée correspond aux matrices M et N ($k^2 + h^2$) quelle que soit la méthode. Pour la méthode matricielle, on ajoute la place mémoire prise par la matrice de passage P et sa transposée soit $2.k.h$ d'où $\eta_m = k^2 + h^2 + 2.k.h$ Pour la méthode vectorielle, on ajoute la place mémoire prise par les vecteurs V^l et V^c, soit $2.h$ D'où $\eta_v = k^2 + h^2 + 2.h$. Il vient donc que :

$$k.h^2 + k^2.h > h^2 \quad \Rightarrow \quad \eta_m > \eta_v$$

$$k^2 + h^2 + 2.h.k > k^2 + h^2 + 2.h \quad \Rightarrow \quad \mu_m > \mu_v$$

Q.E.D.

Proposition 3 : *L'algorithme* A_v *de la méthode vectorielle d'agrégation est partiellement supérieur au sens de* η, *à l'algorithme* A_i *de la méthode intermédiaire.*

Démonstration - Dans le cas intermédiaire, l'algorithme itère selon les dimensions de N. Pour chaque élément de N il somme les éléments de M correspondants en les repérant par rapport aux effectifs des classes précédantes. Pour chaque élément $n_{i,j}$, i itérations seront nécessaires (en ligne) pour se placer sur le premier élément de M puis e_i pour sommer les éléments utiles. Le même raisonnement s'applique en colonne, de sorte que :

$$\eta_i = \sum_{i=1}^{k} \sum_{j=1}^{k} i.e_i.j.e_j \tag{7}$$

Par ailleurs, nous pouvons transformer l'expression $\eta_v = h^2$. En effet

$$h = \sum_{i=1}^{k} e_i$$

$$h^2 = \left(\sum_{i=1}^{k} e_i \right)^2$$

$$h^2 = \sum_{i=1}^{k} e_i \sum_{j=1}^{k} e_j$$

$$\sum_{i=1}^{k} e_i \sum_{j=1}^{k} e_j < \sum_{j=1}^{k} i.e_i.j.e_j$$

Q.E.D.

Proposition 4 : *L'algorithme A_i de la méthode intermédiaire d'agrégation est partiellement supérieur au sens de μ, à l'algorithme A_m de la méthode matricielle.*

Démonstration - Compte tenu de l'hypothèse (**H.1**), on forme $1 < k < h$ d'où $k^2 + h^2 + 2.k < k^2 + h^2 + 2.k.h \Rightarrow \mu_i < \mu_m$.
Q.E.D.

Proposition 5 : *L'algorithme A_i de la méthode intermédiaire d'agrégation est partiellement supérieur au sens de μ, à l'algorithme A_v de la méthode vectorielle.*

Démonstration - Partant de $1 < k < h$, on forme $k^2+h^2+2.k < k^2+h^2+2.h \Rightarrow \mu_i < \mu_v$.
Q.E.D.

REMARQUE : Nous avons pu comparer tous les algorithmes deux à deux - voir le tableau N°2.2.a -, à l'exception de A_m et A_i au sens de μ.

TABLEAU N°2.2.a - Comparaison des performances des méthodes d'agrégation d'une matrice $M(h,h)$ vers une matrice $N(k,k)$

MÉTHODES	NOMBRE D'ITÉRATIONS	PLACE EN MÉMOIRE
Matricielle	$k.h^2 + k^2.h$	$k^2 + h^2 + 2.k.h$
Vectorielle	h^2	$k^2 + h^2 + 2.h$
Intermédiaire	$\sum_{i=1}^{k}\sum_{j=1}^{k} i.e_i.j.e_j$	$k^2 + h^2 + 2.k$

Nous ne pouvons seulement que présumer A_m supérieur à A_i au sens de μ par une simulation des μ_i et μ_m en fonction des k croissants - Fig.22.c.

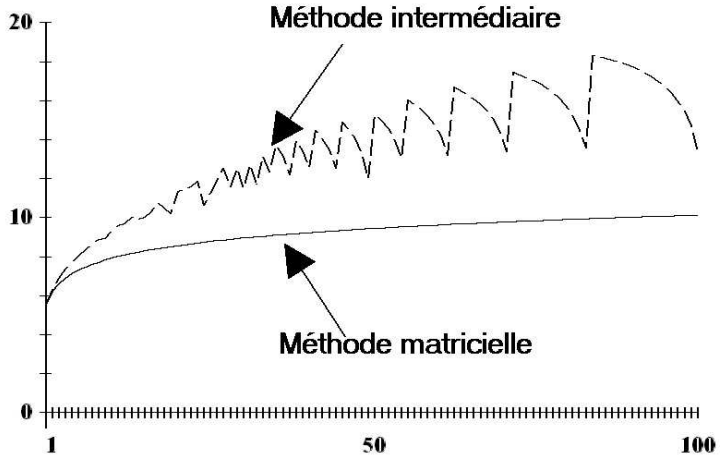


Fig.2.2.c - Comparaison des vitesses de calculs des méthodes matricielles et intermédiaires[35]

[35]- Si l'on suppose $h = 100$ et les classes homogènes et égales à $\frac{h}{k}$, sauf la k-ème qui opère l'ajustement on a $\varepsilon_s = h/k$ avec $s \in [1, k-1]$ et

$$\varepsilon_k = h - \sum_{i=1}^{k-1} e_i$$

Les valeurs des $\mu_i(k)$ et $\mu_m(k)$ étant trop grandes pour être reportées telles quelles dans le graphique, nous sommes passés en logarithme nepérien par la transformation suivante $t(\mu) = Ln(\frac{\mu}{1000})$.

III/ Les Algorithmes Graphiques et les Algorithmes de Communication

L'utilisation d'algorithmes graphiques, quoiqu'elle puisse paraître étrangère à la modélisation économique, peut présenter un intérêt dans le cas de la modélisation multi-régionale. La représentation cartographique peut y tenir un grand rôle. Enfin, nous ne pouvons pas aborder la question des algorithmes de gestion des données sans parler des algorithmes de communication, relatifs à l'échange d'informations entre plusieurs postes au sein d'un réseau.

a) LES ALGORITHMES GRAPHIQUES

La représentation graphique peut être perçue comme une illustration symbolique qui vient à l'appui d'un discours littéraire et/ou formalisé. L'informatique propose des algorithmes assez performants pour "dessiner" des formes intelligibles sur un écran. Cependant, cette seule vision du rôle de la représentation graphique est réductrice. Non seulement parce que chaque individu perçoit de manière spécifique les informations de son environnement[1], mais également parce que la représentation graphique peut faire émerger une information nouvelle. La représentation graphique intervient comme mode de transformation des informations ou comme moyen de simulation en géographie quantitative comme en économie spatiale[2]. La géométrie requiert un langage et une grammaire (algorithmique) spécifique qui permet des raisonnements et des constructions géométriques, comme nous le verrons en présentant les algorithmes de pavage. Enfin, nous verrons la représentation graphique peut également être un moyen de décéler une logique spécifique de développement d'un phénomène ; le développement urbain et la logique fractale.

i - Les procédures graphiques usuelles

Les algorithmes graphiques bien qu'ils aient pour but de fournir une représentation, comportent une "composante" mathématique, en l'occurrence géométrique. Depuis que les ordinateurs proposent un affichage graphique plus élaboré que le simple affichage des caractères textuels, la plupart des langages de programmation ont intégré des fonctions graphiques spécifiques (ex. : BORLAND, Turbo-Pascal Graphix Toolbox, Sèvres, Borland, 1986, 256 p.). On comprend que le rôle de ces procédures soit déterminant pour permettre la représentation de formes (courbes, surfaces, etc.) dans un domaine donné (plan, espace) tout en permettant une manipulation formelle des concepts. Plus concrètement, le problème est lié à l'emploi d'un langage approprié à la représentation graphique sur un ordinateur et aux contraintes matérielles : un écran est composé de pixel - *i.e.* picture element[3]. La pleine exploitation des propriétés graphiques des écrans

[1]- Informations visuelles (graphiques, textuelles etc.) et auditives ne sera pas mémorisé de la même manière selon les individus - voir M.ROBERT ("Le traitement de l'information", in G.WILLET (ED.), *La communication modélisée - une introduction aux concepts, aux modèles et aux théories*, Ottawa, Ed. du Renouveau pédagogique, pp.198-222).

[2]- Voir à ce propos S.RIMBERT (*Carto-graphies*, Paris, Hermès, Coll., Traité des nouvelles technologies, 1990, 175 p.).

[3]- Rappelons la distinction entre 1) Le dessin pixel par pixel - *i.e.* le logiciel graphique lit un fichier graphique qui lui fournit les couleurs de chaque point de l'image, et affiche celle-ci en balayant l'écran du point de coordonnées (1,1) jusqu'à celui de coordonnées (768,1024). 2

de micro-ordinateurs en modélisation n'est pas très ancienne[4]. La plupart des langages actuels fournissent des outils de représentation graphiques élémentaires (citons par exemple C.DUVAL, 1989 pour le langage C et R.DONY, 1991 pour le langage Pascal).

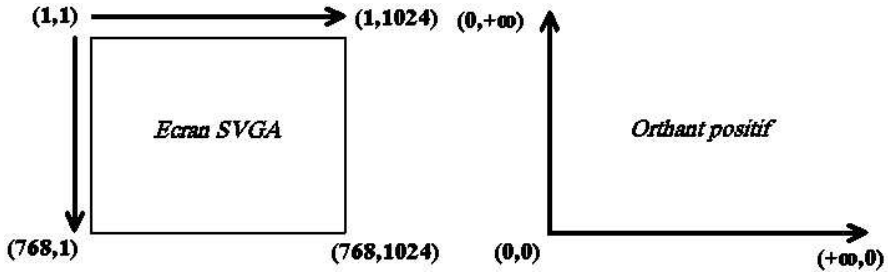


Fig.20 - Comparaison de l'affichage graphique et de la représentation mathématique

Les étapes de la programmation graphique sont sensiblement les mêmes d'un langage à un autre. En premier lieu il s'agit de vérifier les problèmes de distorsion graphique (un objet "cercle" est affiché comme une "ellipse") - voir R.DONY (*op.cit.*). Ensuite il faut définir la fenêtre de représentation afin de savoir quels seront les points que l'on pourra afficher - la définition des points par le système n'est jamais conforme à celle, par exemple, d'un orthant nord-est mathématique - Fig.20.

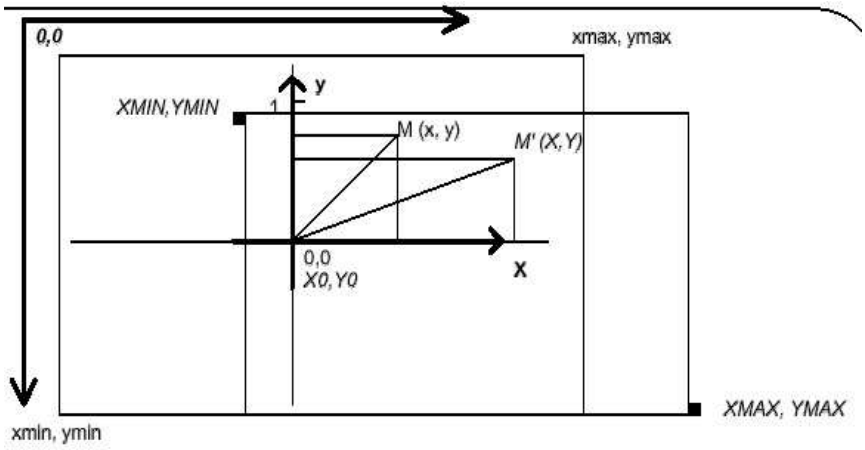


Fig.21 - Croisement des deux repères graphiques[5]

- Le dessin vectoriel - *i.e.* le logiciel graphique exécute les instructions fournies dans un fichier graphique telles que "tracer une droite de telle épaisseur de tel point à tel autre point", ou bien "remplir de telle couleur le polygône ayant les coordonnées...".

[4]- Avant les années 80, les graphiques étaient traçés avec des symboles textuels (*,+,- etc.) - voir R.A.GUEDJ & H.A.TUCKER (1979) à propos des premières fonctions graphiques FORTRAN. Les écrans ont progressivement atteint des tailles confortables (CGA : 200x640, VGA : 640x480, SVGA : 1024x480 puis 1280x1024).

[5]- D'après J.FRUITET (Infographie, *Mimeo*, Université de Marne-la-Vallée, 1995, p.7).

La représentation graphique implique un changement d'échelle éventuel. Dans ce cas, il suffit en effet de faire correspondre les points extrêmes du graphique à représenter avec ceux de la fenêtre d'affichage de l'écran - un paramétrage différent peut toutefois conduire à des effets de zoom. Plus généralement, il faut donc passer des coordonnées d'un point du plan vers celles relatives à l'écran comme suit :

$$\begin{cases} X = \frac{(x-x_{min})(X_{max}-X_{min})}{x_{max}-x_{min}} + X_{min} \\ Y = \frac{(y-y_{min})(Y_{max}-Y_{min})}{y_{max}-y_{min}} + Y_{min} \end{cases}$$

$$\Rightarrow \begin{cases} X_{cran} = \frac{(x-x_{min})(X_{max}-X_{min})}{x_{max}-x_{min}} + X_{min} \\ Y_{cran} = Y_{max} - \left(\frac{(y-y_{min})(Y_{max}-Y_{min})}{y_{max}-y_{min}} + Y_{min} \right) \end{cases}$$

Une application classique en modélisation économique est celle de l'économie régionale, à savoir la représentation cartographique de données ou de résultats de simulation (A.DAUPHINÉ, 1987).

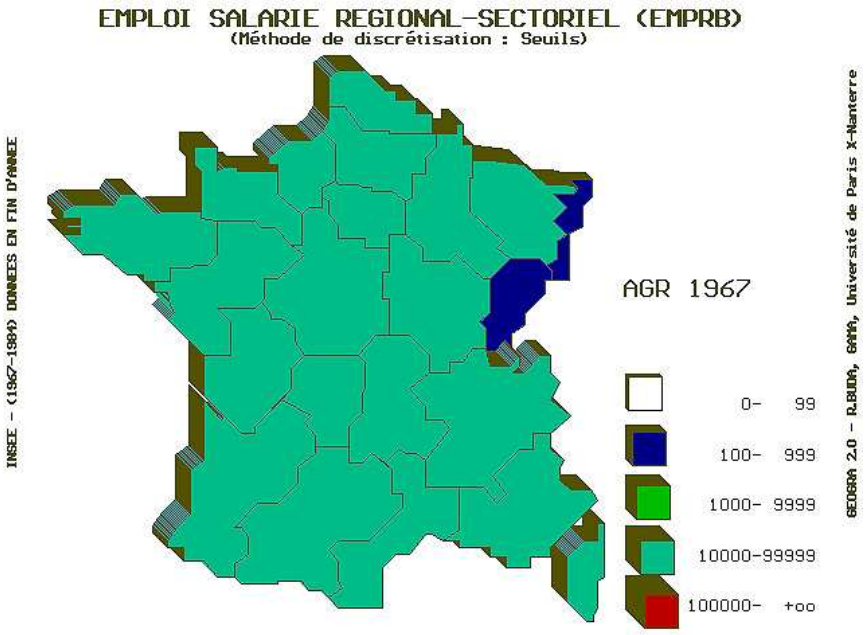


Fig.22 - Construction de régions par l'instruction polygône avec le module GEOGRA[6]

[6]- D'après notre module cartographique GEOGRA (note de travail, 1992). Les coordonnées cartographiques des régions françaises sont stockées dans un fichier. Le logiciel trace ensuite un polygône pour chaque région, le choix de la couleur de remplissage dépend alors du fichier de données numériques. Un effet de relief est également proposé pour mettre en évidence des régions en raison de l'importance de leurs valeurs numériques.

Il s'agit donc de visualiser des grandeurs sur un espace déterminé découpé en entités (un pays en régions par exemple). Les logiciels cartographiques recourent donc à des procédures de tracé (droite, polygone, remplissage etc.) - Fig. 22 - , mais également à des algorithmes de discrétisation des données. Il s'agit de diviser en classes l'échantillon de données à cartographier.

Plusieurs méthodes existent[7] :

1 - les discrétisations intuitives ou arbitraires (liées à l'intuition de l'auteur),

2 - les discrétisations exogènes (liées à des valeurs connexes à celles qu'il s'agit de cartographier),

3 - les discrétisations mathématiques. Citons la méthode des égales étendues : Si V_{max} et V_{min} (resp.) sont les valeurs maximale et minimale (resp.), c le nombre de classes souhaité, alors on appelle étendue la valeur

$$e = \frac{V_{max} - V_{min}}{c}$$

Une valeur v de l'échantillon à cartographier appartient à la classe i si elle vérifie la relation :

$$V_{min} + (i-1).e \leq v < V_{min} + i.e$$

La méthode des progressions arithmétiques et celle des progressions géométriques sont des généralisations de la première. Dans le premier cas, le facteur de progression est

$$X = \frac{V_{max} - V_{min}}{\sum_{i=1}^{c} i}$$

et la relation à vérifier est du type

$$V_{min} + (i-1).X \leq v < V_{min} + i.X$$

dans le second cas, on a

$$X^c = \frac{V_{max}}{V_{min}}$$

et la relation à vérifier est du type

$$V_{min}.X^{i-1} \leq v < V_{min}.X^i$$

4 - les discrétisations statistiques et probabilistes (selon la moyenne et l'écart-type, ou les quantiles, ou les classes équiprobables etc.),

5 - les discrétisations graphiques. Citons la méthode des seuils observés, où l'on découpe l'échantillon en suivant une distribution de fréquence des données.

6 - les discrétisations expérimentales (liées à des seuils théoriques tels que la loi rang-dimension de Zipf).

[7]- Voir les développements dans C.CAUVIN, H.REYMOND et A.SERRADJ (*Discrétisation et représentation cartographique*, Montpellier, Reclus, 1987, 116 p. + Programmes).

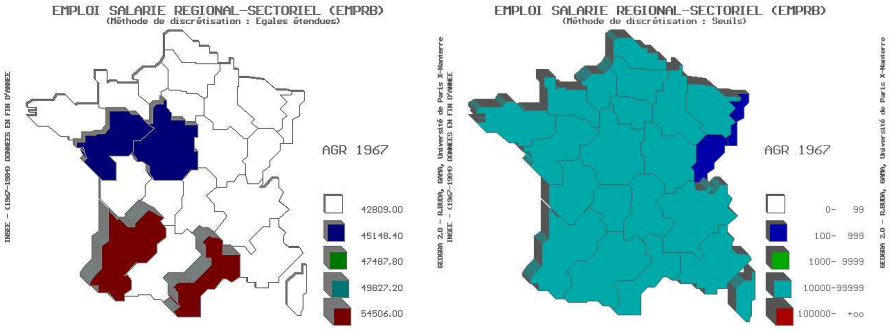


Fig.23 - Affichage d'une même série par deux méthodes de discrétisation avec GEOGRA

Certains effets de représentation peuvent être recherchés, tels que la détermination de faces cachées en graphisme 3D. Nous devons mentionner l'Algorithme du peintre (M.E.NEWELL, R.G.NEWELL et T.L.SANCHA, 1972) qui permet de représenter des volumes dans l'espace, à partir des coordonnées d'un observateur et d'une liste d'ordre des priorités d'affichage des volumes selon leur éloignement par rapport à l'observateur (R.DONY, *op.cit.*, 1991, pp.331-61 et *op.cit.*, 1986, pp.21-92). Cet algorithme de modélisation 3D, qui permet la génération de paysage virtuel[8] permettrait ainsi de disposer de simulations de développement économique avec des conséquences économiques et écologiques - certains algorithmes simulent le développement des végétaux.

ii - Les algorithmes de maillage, de pavage et de construction d'espaces

La représentation cartographique peut avoir une utilité synthétique, notamment lors de la construction de la matrice de contiguïté (Cf.Supra, §II-b-ii). Chaque région est symbolisée par un point et des traits marquent les contiguïtés entre régions ; si les régions i et j sont contigues, alors les éléments $[i, j]$ et $[j, i]$ seront égaux à 1, 0 sinon - Fig.24.

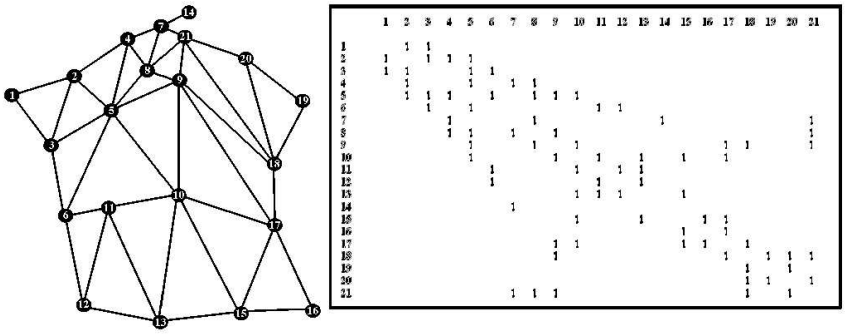


Fig.24 - Passage du graphe à la matrice des contiguïtés[9]

[8]- Voir X.G.VIENNOT (1999) et I.DANIALA et al. (2003, pp.170-91). Au sujet de logiciel de modélisation 3D, voir B.JOLIVALT (*Graphisme 3D avec Bryce 4*, Paris, Campuspress, 1999, pp.221-58).

[9]- Quand i et j ont une frontière commune, la matrice de continguïté comportera le nombre 1 à la iè ligne, jè colonne et symétriquement (j, i) (J.R.BOUDEVILLE, *Aménagement du*

La géométrie algorithmique peut aider à combiner les précédents algorithmes aux résultats de la géométrie[10]. Il s'agit d'une branche des mathématiques qui se propose de formaliser la démarche de construction des objets propres à la géométrie - en plus des domaines d'implémentation habituels tels que l'ensemble booléen, l'ensemble des réels, des entiers, voire des complexes etc., le domaine des algorithmes géométriques s'élargira au plan et à l'espace. On trouve ainsi des notions spécifiques telles que le balayage - *i.e.* le déplacement d'un élément d'un point vers un autre selon une direction déterminée[11]. La notion de cloisonnement, consiste à subdiviser avec des segments, le domaine d'étude en régions élémentaires trapézoïdales ; le nombre d'arêtes dépend du nombre de segments ainsi que de leurs intersections (*Ibid.*, pp.42-47). A partir de ces éléments, on peut modéliser ce que l'on appelle le maillage - *i.e.* la construction de surfaces par interpolation sur des arêtes de polygones (à propos des algorithmes incrémentaux et des algorithmes par balayage (*Ibid.*, pp.275-94 ; D.BEAUQUIER et al., 1992, pp.379-408 ; I.DANAILA et al., 2003, pp.193-232). Ce type de modélisation est requis pour représenter des systèmes différentiels, par exemple le problème de la taille du maillage est au coeur du problème des prévisions météorologiques[12].

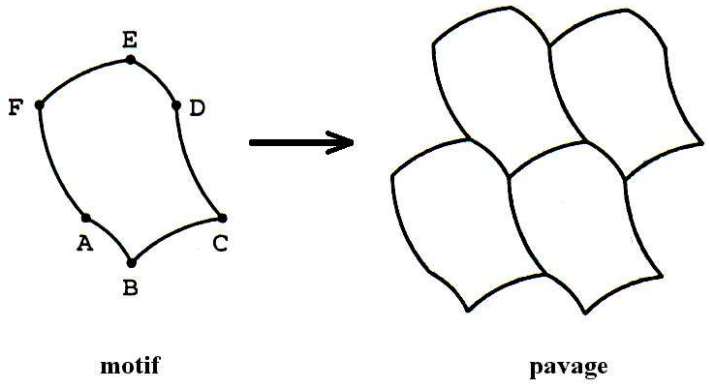


Fig.25 - Du motif au pavage

Les algorithmes de pavage, quant à eux, consistent, à créer un motif géométrique - Fig.25 - pour le multiplier et en remplir une surface déterminée - voir les programmes en CAML (langage développé à l'INRIA) de G.COUSINEAU et M.MAUNY (1995, pp.301-56).

---

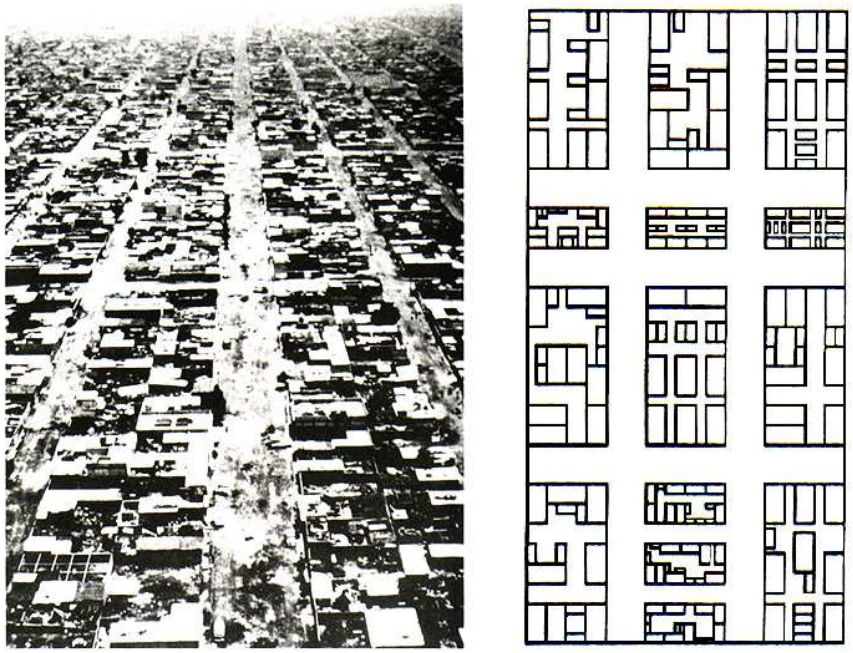
territoire et polarisation, Paris, Litec, 1972, pp.26-27).

[10]- Certaines propriétés géométriques doivent être connues pour éviter des erreurs d'interprétation. Citons la conjecture de Guthrie, connue sous le nom de "Théorème des quatre couleurs" - bien qu'il n'ait pas été démontré formellement (K. Appel, W. Haken. "Every planar map is four colourable", *Contemporary Mathematics*, N°98, 1989) - : il est possible de colorier n'importe quelle carte avec quatre couleurs en évitant que deux régions voisines aient la même couleur.

[11]- Soit Y structure appelée état du balayage et X structure appelée suite des événements, les informations contenues dans Y sont liées à la position de la droite de balayage et évoluent lorsque celle-ci se déplace. Cette structure n'est mise à jour que lorsque la droite de balayage passe par un nombre fini de positions discrètes appelés événements (J.D.BOISSONNAT & M.YVINEC, 1995, pp.37-41).

[12]- M.ROCHAS et J.P.JAVELLE, *La météorologie - prévision numérique du temps et du climat*, Paris, Syros, 1993, pp.81-124.

Utilisés notamment en modélisation spatiale, les objets fractals et la logique floue ne constituent pas à proprement parler des algorithmes. Les objets fractals (B.MANDELBROT, 1977) sont des objets géométriques qui présentent la propriété de comporter une homothétie interne. L'objet reste semblable quelle que soit l'échelle d'observation. P.FRANKHAUSER[13] a obtenu des résultats intéressants en reconstituant des quartiers urbains à partir de ce mode de construction géométrique - tiré de l'ouvrage, Fig.26.



Ville Satellite de Mexico
National Geographic (11/1973)

Fractale à base de rectangles
P.FRANKHAUSER (op.cit., p.161)

Fig.26 - Comparaison entre la photo d'un quartier et la synthèse fractale

La Logique floue (L.A.ZADEH, 1965), enfin, consiste à reprendre dans la théorie des ensembles, la relation d'appartenance. Soit un ensemble A, dans la théorie classique des ensembles, on définit une fonction $\psi_A : X \rightarrow \{0,1\}$ - si $X \in A$ alors $\psi_A = 1$ sinon $\psi_A = 0$. Dans la théorie des sous-ensembles flous, on a $\psi_A : X \rightarrow [0,1]$ - les valeurs 0 et 1 apparaissent comme des cas particuliers entre lesquels on peut avoir une infinité de cas intermédiaires, par ex. $\psi_A = 0.3333$ où X appartient à A avec un certain degré 0.3333. Ce mode de représentation ensembliste a permis, d'une part d'éprouver des méthodes de classification alternatives à celles de l'analyse de données[14], d'autre part d'enrichir les analyses régionales et urbaines[15].

[13]- *La fractalité des structures urbaines*, Paris, Anthropos, Coll.Villes, 1994, 291 p. Pour des programmes d'objets fractals voir G.LÉVY (1994, pp.413-57), ainsi que R.DONY (*op.cit.*, 1991, pp.151-78).

[14]- Voir P.TRAN QUI (*Les régions économiques floues - application au cas de la France*, Dijon, Librairie de l'université, Coll. IME, 1978, 159 p.), à propos d'un redécoupage régionale économiquement pertinent.

[15]- Voir C.PONSARD et B.FUSTIER (EDS) (*Fuzzy Economics and Spatial Analysis*, Dijon, Librairie de l'université, Coll. IME, 1986), à propos du calcul des temps de transport.

b) LES ALGORITHMES DE COMMUNICATION

Il peut paraître curieux de s'intéresser aux algorithmes de communication dans le cadre de la modélisation économique. Pourtant, rappelons que les ordinateurs ont été conçus dès le départ, pour communiquer entre eux. D'autre part, bien que la modélisation économique nous ait habitués à une configuration du type un décideur-un système, pourquoi ne pas en envisager d'autres, telles que celles proposées par les jeux d'entreprises (plusieurs décideurs-un système) ? A condition de savoir très exactement où il évolue (simulation, macroéconomie, microéconomie, expérimentation etc.), c'est une condition essentielle de l'interprétation pertinente des résultats, le modélisateur peut parfaitement utiliser les réseaux neuronaux, la programmation parallèle[16] etc. Nous présenterons, de manière sommaire et non exhaustive, les éléments de base nécessaires à la compréhension de la programmation des algorithmes sur réseaux, puis nous évoquerons les applications qui ont été proposées.

i - Réseaux locaux, protocoles et communication entre ordinateurs

On parle de réseau informatique local (en anglais LAN - *i.e.* Local Area Network) lorsqu'au moins deux ordinateurs sont reliés pour échanger des données et/ou exécuter des programmes sur l'un à partir de l'autre.

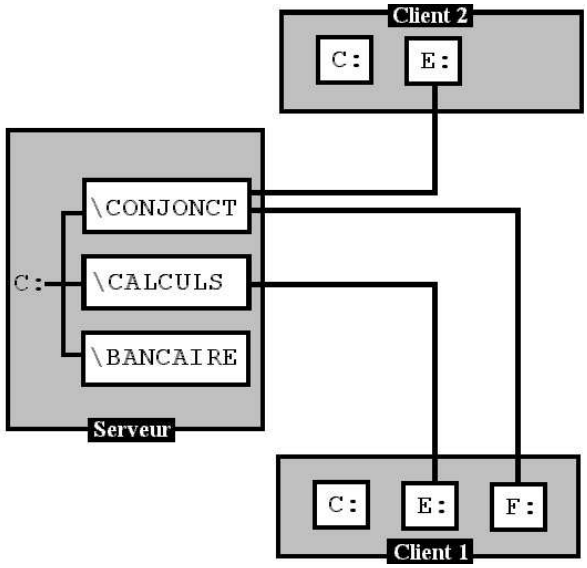


Fig.27 - Architecture clients/serveur[17]

[16]- Citons les résultats obtenus en avril 2000 en matière de décryptage de clé publique grâce à la contribution bénévole de 9500 internautes qui ont laisser calculer sur leur ordinateur le programme `ecdl2K-108`.

[17]- Ici, le serveur dispose de données de conjoncture (répertoire `C :\CONJONCT`), bancaires (répertoire `C :\BANCAIRE`) et de logiciels de modélisation (répertoire `C :\CALCULS`). Les clients N°1 et N°2 ont accès aux données de conjoncture (`E :\CONJONCT`), tandis que seul le N°1 a accès aux logiciels (`F :\CALCULS`). Aucun d'eux n'a accès aux données bancaires.

L'intérêt de la connection simultanée de plusieurs ordinateurs à un même système (le réseau local ou InterNet) tient dans l'accès pour tous à des ressources partagées. Même si dans la pratique, cet accès peut éventuellement être restreint pour certains et intégral pour d'autres selon la politique réseau suivie par l'administrateur réseau. Ceci explique que l'un des postes dispose d'un statut prépondérant, le serveur, par rapport aux autres, les clients - Fig.27. Au delà de deux ordinateurs connectés - ce qui est peu intéressant - il existe plusieurs configurations de réseaux. Citons principalement, le réseau complet (tous les opérateurs sont connectés les uns avec les autres), le réseau en anneau (chaque opérateur est relié à deux autres opérateurs), le réseau en étoile (un opérateur est relié à tous les autres), le réseau en arbre etc.

Le premier problème à résoudre est l'acheminement dans de bonnes conditions d'un message (un ordre, un fichier etc.) de l'émetteur vers le récepteur (routage) sans un blocage des communications (collision). Il existe plusieurs principes de routage (A.TANENBAUM, 1996, pp.364-93).

Décrivons d'abord le Principe du Token ring - *i.e.* l'anneau à jeton - Fig.28. Quatre postes (A,B,C et D) sont connectés à un réseau en anneau. Tant que personne ne souhaite envoyer un message, un jeton circule de poste en poste par l'intermédiaire du réseau. A l'étape (1), le jeton est rejeté par B qui ne souhaite pas envoyer de message. C reçoit le jeton (2), et comme il souhaite envoyer un message à A, il garde le jeton et expédie le message sur le réseau (3). D reçoit le message qui ne lui est pas destiné (4), et le réinjecte alors sur le réseau (5). Finalement, A reçoit alors le message (6). A copie le message en mémoire centrale et envoie un accusé réception (a.r.) à C (7). L'a.r. parvient à B qui n'est pas le destinataire (8). Celui-ci le réinjecte dans le réseau (9). Finalement l'a.r. parvient à C (10), lequel informé que le message est bien arrivé peut réinjecter le jeton dans le réseau (11) ; le jeton parvient alors à D... et ainsi de suite. On peut également appliquer la Politique d'accès CSMA/CD - *i.e.* Carry Sense Multiple Acess with Collision Detection. Supposons cinq postes (A,B,C,D et E) parmi lesquels deux souhaitent envoyer un message (C à B et E à D). Par hasard, C et E se mettent à l'écoute du réseau, où rien n'est émis. Par hasard, ils émettent alors en même temps. Leurs signaux se mélangent et se brouillent. C et E écoutent le réseau et se rendent compte du brouillage, si bien qu'ils cessent d'émettre. Après un laps de temps d'attente aléatoire pour C et E, C se remet à l'écoute du réseau le premier. La voie étant libre, C se remet à émettre à destination de B. Tous les postes sont à l'écoute, mais seul B copie le message puisqu'il lui est destiné. Ajoutons que dans certains types de réseaux (en particulier Internet), les algorithmes de routage doivent déterminer le chemins le plus court. Nous ne donnerons pas d'exemple d'algorithmes de routage, dans la mesure où ils rappellent ceux de la recherche opérationnelle évoquée plus haut, ni de programmes trop fortement connotés programmation système[18].

[18]- Nous invitons le lecteur à consulter notamment J.BEAUQUIER et B.BÉRARD (*op.cit.*, pp.417-33) pour les algorithmes et T.BASTIANELLI & G.LEBLANC (1990) pour des programmes systèmes écrits en langage C.

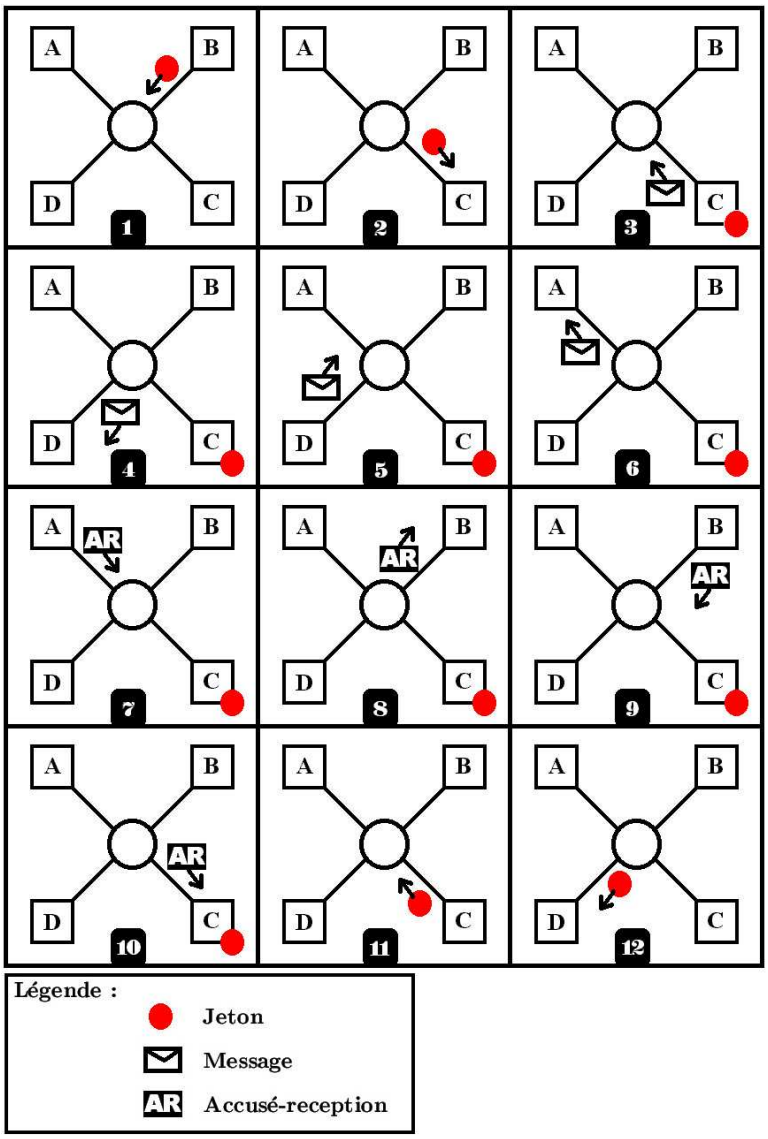


Fig.28 - Principe de l'anneau à jeton

Le second problème qu'il convient de traiter par une algorithmique appropriée est celui de la synchronisation des processus. En d'autres termes, au moins deux clients cherchent à accéder simultanément à une ressource qui ne peut être utilisée que par un seul poste à la fois[19]. Deux informaticiens, T.J. DEKKER[20] puis G.L PETERSON (1981), ont proposé une solution, appelée Algorithme

[19]- En réalité les choses sont un peu plus compliquées : deux processus (par ex. deux programmes) détiennent chacun deux ressources (par ex. des données en mémoire). Si chacun demande une ressource détenue par l'autre il existe un risque d'attente indéfinie appellée étreinte fatale, lorsque les ressources ne peuvent être partagées.

[20]- Décrit dans E.W. DIJKSTRA, *Cooperating Sequential Processes*, Eindhoven Univ. Of Technology, EWD 123, 1965. Voir également E.W. DIJKSTRA, "Solutions of a Problem in Concurrent Programming Control", *CACM*, 8(9), 1965, 569 p.

de Dekker-Peterson[21]. Chaque "intervenant", processus, dispose d'un drapeau, flag, qui teste la disponibilité de la ressource qui doit être sollicitée. Le drapeau de chaque processus est en position "mis" ou "levé" (à la manière analogue aux feux tricolores qui indiquent si un véhicule peut passer ou non). L'accès à la ressource pour l'un l'exclut pour les autres (quand le feu est vert pour une voie, il est rouge pour les autres). Nous en verrons une illustration sur un exemple plus concret - Cf. Infra §III-b-ii.

```
PROCESSUS N°1                          PROCESSUS N°2
Début                                  Début
Flag_1=mis                             Flag_2=mis
Tour=2                                 Tour=1
Tant que                               Tant que
   (Flag_2=mis et que Tour=2)             (Flag_1=mis et que Tour=1)
   alors Attente active                   alors Attente active
{ Section critique }                   { Section critique }
Flag_1=lev                             Flag_2=lev
Fin                                    Fin
```

Fig.29 - Algorithme de Dekker-Peterson[22]

Citons également l'Algorithme du sémaphore de E.W.DIJKSTRA (1968), basé plutôt sur des signaux "calculés" (les sémaphores).

```
P(S) - Tentative d'entrée              V(S) - Libération de la place
Début                                  Début
  e(S)=e(S)-1 ;                          e(S)=e(S)+1 ;
  si (e(S)<0) alors                      si (e(S)<=0) alors
     état(r)=bloqué ;                       sortir le processus q de f(S) ;
     mettre r dans la file f(S) ;           état(q)=actif ;
  fin de si ;                            fin de si
Fin ;                                  Fin
```

Fig.30 - Algorithme du sémaphore

où un sémaphore S est constitué d'une variable entière $e(S)$ et d'une file d'attente $f(S)$. A sa création, la file est vide et $e(S)$ est initialisé à une valeur entière non négative $e_0(S)$. Deux opérations indivisibles et exclusives permettent d'agir sur le sémaphore[23] - Fig.30. Il faut également mentionner l'algorithme de L.LAMPORT (1978) qui utilise également une file d'attente pour gérer les exclusions[24]. En tout état de cause, l'implémentation de systèmes utilisant le réseau renvoie à l'ensemble des problèmes relatifs à la programmation en parallèle. En particulier au choix du langage de programmation, qui n'est pas anodin (ADA, Algol, Concurrent Pascal etc.). Ce choix dépend en particulier de la nature du

[21]- Pour un exposé sommaire, mais plus complet, voir T.FALISSARD (*Le logiciel système*, Paris, PUF, Coll.Que sais-je, pp.44-50).

[22]- D'après T.FALISSARD (*Ibid.*).

[23]- "*Un sémaphore peut être comparé à un local ayant une entrée et une sortie, qui peut accueillir $e_0(S)$ personnes au plus. P correspond à une tentative d'entrée : soit elle réussit, s'il reste de la place dans le local ($e(S) > 0$) et une place est prise ($e(S)$ diminue de 1) ; soit elle échoue, quand le local est plein, et il faut attendre à l'entrée. V correspond à la libération d'une place dans le local ($e(S)$ augmente de 1) et il faut faire entrer une personne en attente à l'entrée, s'il y en a ($e(S) \leq 0$)*", d'après J.LONGCHAMP (*op.cit.*, 113).

[24]- Voir à ce propos J.BEAUQUIER et B.BÉRARD (*op.cit.*, pp.437-86).

parallélisme (lectures simultanées autorisées ou non et/ou écritures simultanées autorisées ou non) - voir les explications de T.CORMEN et al. (*op.cit.*, 1994, pp.675-715), le cadrage linguistique de J.LONGCHAMP (1989, pp.101-30) ainsi que les algorithmes de S.BAASE (1988, pp.361-95).

ii - Les applications économiques sur réseaux locaux et sur internet

L'examen des thèmes de recherches actuels montre que ce sont essentiellement les économistes spécialistes d'économie expérimentale et les économistes issus de la branche dite de la Computational Economics[25] qui se sont intéressés au problème des réseaux locaux et par extension à Internet. Les premiers utilisent les réseaux comme outils d'expérimentation[26], tandis que les seconds considèrent les réseaux comme un objet de recherche. Dans un cas comme dans l'autre, c'est la nécessité de coller davantage à la réalité, qui a déterminé les choix instrumentaux. Pour l'économie expérimentale, le problème est de collecter les informations de la manière la plus fiable possible (sans biais ni délai) ; pour la Computational Economics, le problème est de se rapprocher le plus possible de la complexité des phénomènes[27]. Certes, le thème des réseaux dans la recherche en matière d'économie de l'information et des télécommunications n'est pas nouveau[28], mais les approches intégrant la démarche algorithmique et la programmation réseaux sont quant à elles novatrices.

Expérimentation et échanges d'informations - La première exploitation des réseaux locaux dans le cadre des recherches en science économique date de 1976 avec les travaux d'A.H.WILLIAMS[29]. Il s'agissait de mettre au point une expérimentation économique dont les informations devaient transiter par les ordinateurs. Depuis d'autres programmes ont été développés citons en 1994 le logiciel ESL Double Auctions basé sur le principe de la double enchère. Ce logiciel, comme tous ceux ayant pour but de gérer un marché, est confronté au problème de la gestion optimale des ordres. Ainsi, pour notre part avec le logiciel ECHANGE que nous avons développé (voir notre note de travail 1999.b), les opérateurs passent des ordres d'achat et de vente sur plusieurs marchandises[30].

[25]- La publication en 1996 du volume *Handbook of Computational Economics* dans la célèbre collection des Handbook des éditions Elsevier, indique un intérêt croissant d'une partie de la communauté des économistes pour l'intégration de l'algorithmique dans la théorie économique. Pour ses promoteurs, et en particulier pour les fondateurs de la toute jeune Society of Computational Economics (1994), il ne fait aucun doute que les raisonnements algorithmiques sont consubstantiels des raisonnements économiques : "[...] *it will be as difficult to do economics without having a good understanding of computing hardware and software and numerical methods as it is to do economics without a solid understanding of real analysis, probability and statistics.*" (*Ibid.*, pp.viii).

[26]- Voir à ce propos MACKIE-MASON J.K., VARIAN H.R. ("Some Economics of the Internet", *Working Paper*, University of Michigan, Nov., 1992) ainsi que MACKIE-MASON J.K., VARIAN H.R. ("Pricing the Internet", *Working Paper* , University of Michigan, Apr., 1993).

[27]- J.RUST "Dealing with Complexity of Economic Calculations", *Working Paper*, Yale University, oct. 1997.

[28]- Citons notamment C.ANTONELLI (ED.), *The Economics and Information Networks*, Amsterdam, North- Holland, 1992, 477 p.

[29]- WILLIAMS A.H., "Computerized Double Auction Markets : Some Initial Experimental Results", *Journal of Business*, Vol.53, 1980, pp.235-58. Voir également notre note de travail (2000.a).

[30]- C'est de ce type de procédure, que des algorithmes tels que celui de Dekker-Peterson

A l'Étape N°1 - Fig.31 -, les opérateurs rédigent leur annonce (offre) ce qui se traduit par l'envoi sur le serveur d'un fichier - voir légende Fig.31. Ce dernier est en attente symbolisée par une boucle de lecture (puisqu'il attend que tous les fichiers d'offre lui soient parvenus). A l'Étape N°2, l'opérateur K n'a pas encore rédigé son offre, et les autres opérateurs sont en attente (boucle de lecture pour attendre un fichier d'annonce qui n'est pas encore créé). A l'Étape N°3, l'opérateur K envoie son annonce grâce à laquelle, Étape N°4, le serveur peut créer le fichier d'annonces à partir de tous les ordres collectés. A l'Étape N°5, les fichiers d'annonces sont envoyés sur chaque poste, pour y être affiché, Étape N°6.

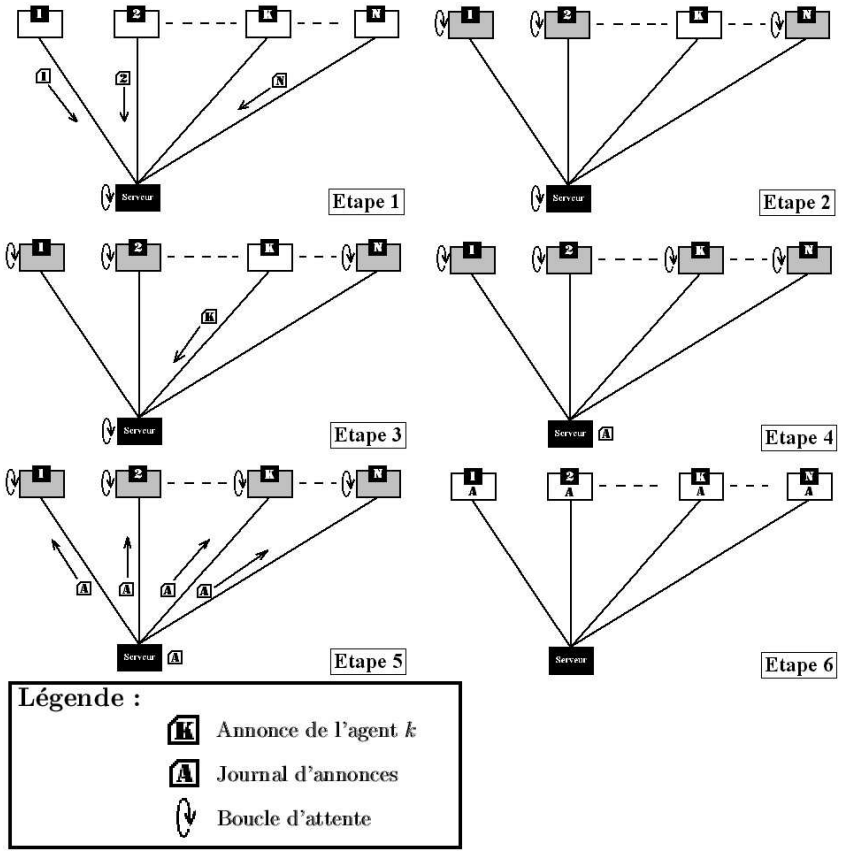


Fig.31 - Collecte des ordres d'achat/vente avec le logiciel ECHANGE

Agent-Based Computational Economics - Avec un même souci de coller à la réalité, certains chercheurs ont proposé des modèles de simulation de marché où chaque opérateur est représenté individuellement. Cette démarché est adoptée dans une optique de jeu évolutionniste. Citons le modèle Trade Network Game[31] développé en 1999 D.McFADZEAN et L.TESFATSION. Il s'agit d'un modèle de simulation qui génère des transactions entre agents, basés sur

interviennent pour éviter qu'un même fichier soit lu ou modifié simultanément par deux postes différents.

[31]- Voir à ce propos D.McFADZEAN and L.TESFATSION (1999), L.TESFATSION (1997) ainsi que D.McFADZEAN (1995).

le principe du dilemme du prisonnier et qui permet d'observer les stratégies et les choix opérés par les agents. A.NAGURNEY (1996) simule une économie multirégionale-multisectorielle sur un système en parallèle et constate le comportement différents de certains algorithmes selon qu'ils sont en parallèle ou non.

Vers des algorithmes de marché - La combinaison des recherches en modélisation et en expérimentation, qui aurait recours à la programmation réseaux, nous semble fructueuse. Les modèles fonctionnent en général sur la base d'hypothèses que les modélisateurs ont le plus grand mal à justifier a posteriori. Nous proposons donc, dans le cadre de notre modèle SINGUL (note de travail 2000.b), de substituer aux hypothèses, des résultats d'expérimentation (ECHANGE) - Fig.32. Ajoutons que la vision algorithmique des mécanismes de marché nous paraît souhaitable pour mieux en comprendre le fonctionnement - voir nos propositions d'Algorithmes de marché[32] en Annexe 3.

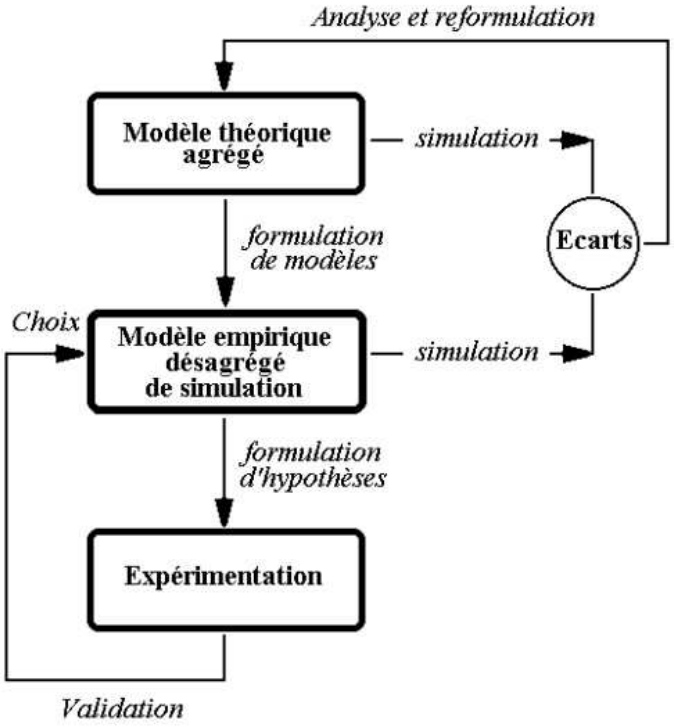


Fig.32 - Combinaison des méthodes expérimentale et de simulation avec le couple de logiciels SINGUL-ECHANGE

[32]- Des formalisations théoriques fondamentales déjà ont été proposées, telle que celle de l'Algorithme de Scarf fondé sur la résolution de système dite Méthode du point fixe - voir H.SCARF (1967) - et à la base du développement actuel des Modèles d'Équlibre Général Calculable. Ils restent de par leur fort degré de désagrégation tributaire d'estimation économétrique.

Le caractère processionnaire ou parallèle des algorithmes de résolution de marché soulève plusieurs questions théoriques, en particulier sur l'efficacité des transactions. En d'autres termes, pour un agent i donné, la qualité de ses transactions (en termes d'utilité) sera-t-elle meilleure s'il cherche par lui-même ses partenaires, ou bien sera-t-elle significativement meilleure si un Planificateur omniscient[33] cherche à sa place ? Par ailleurs, ce raisonnement doit être tenu pour l'ensemble des agents du marché[34].

RÉFÉRENCES

AMMAN H.M., KENDRICK D.A., RUST J. (EDS), (1996), *Handbook of Computational Economics*, Amsterdam, North-Holland, 827 p.

BAASE S., (1988), *Computer Algorithms - Introduction to Design and Analysis*, Reading (Mass.), Addison-Wesley, 415 p.

BASTIANELLI T., LEBLANC G., (1990), *Programmation réseau sur IBM PC et compatibles*, Paris, Eyrolles, 167 p.

BEAUQUIER D., BERSTEL J., CHRÉTIENNE P., (1992), *Éléments d'algorithmique*, Paris, Masson, Coll.Manuels informatiques, 463 p.

BEAUQUIER J., BÉRARD B., (1994), *Systèmes d'exploitation - concepts et algorithmes*, Paris, Edisciences, Coll.Informatique, 541 p.

BOISSONNAT J.D., YVINEC M., (1995), *Géométrie algorithmique*, Paris, Ediscience, Coll. Informatique, 540 p.

CORMEN T., LEISERSON C., RIVEST R., (1994), *Introduction à l'algorithmique*, Paris, Dunod, Coll. 2ème Cycle universitaire/Ecoles d'ingénieurs, 1019 p.

COUSINEAU G., MAUNY M., (1995), *Approche fonctionnelle de la programmation*, Paris, Ediscience, Coll.Informatique, 428 p.

DANAILA I., HECHT F., PIRONNEAU O., (2003), *Simulation numérique en C++*, Paris, Dunod, Coll. Science Sup., 340 p.

DAUPHINÉ A., (1987), *Les modèles de simulation en géographie*, Paris, Economica, 187 p.

DIJKSTRA E.W., (1968), "The Structure of THE Multiprogramming System", *Communications of the ACM*, N°11(5), pp.341-46.

DONY R., (1986), *Calcul des parties cachées - approximations des courbes par la méthode de Bezier et des B-Splines*, Paris, Masson, Coll. Méthodes+Programmes, 238 p.

DONY R., (1991), *Graphisme dans le plan et dans l'espace avec turbo pascal*, Paris, Masson, 380 p.

DUVAL C., (1989), *Graphiques en Turbo C - Techniques de programmation*, Paris, Eyrolles, 354 p.

GUEDJ R.A., TUCKER H.A. (EDS), (1979), *Methodology in Computer Graphics*, Amsterdam, North- Holland, 206 p.

LAMPORT L., (1978), "Time, Clocks and the Ordering of Events in a Distributed System", *Communications of the ACM*, N°21(7), juil. pp.558-65.

LÉVY G., (1994), *Algorithmique combinatoire - méthodes constructives*, Paris, Dunod, Coll.Science informatique, 502 p. + Programmes.

LONGCHAMP J., (1989), *Les langages de programmation - concepts essentiels*, évolution et classification, Paris, Masson, Coll. Manuels informatiques, 239 p.

MANDELBROT B., (1977), *The Fracals*, San Fransisco, Freeman.

[33]- M.POLANYI (1951, *The Logic of Liberty*, Chicago, University of Chicago Press, (trad.1989, PUF, pp.148- 77) explique comment, humainement cette hypothèse n'est pas tenable, même si le Planificateur dispose d'une organisation humaine très hiérarchisée.

[34]- Dans le modèle SINGUL, les agents effectuent leurs transactions en dehors du prix d'équilibre. Les transactions n'ont lieu que si les partenaires ont des prix désirés compatibles, ce qui implique que toutes les transactions sont satisfaisantes pour les agents qui ont contracté. Par conséquent, la situation en dehors de l'équilibre est meilleure (sans être nécessairement optimale) que la situation d'équilibre. Il faut toutefois étudier (par exemple expérimentalement) la modification du comportement des agents s'ils sont informés du prix d'équilibre de leur marché ; rien ne dit que tous modifieront leur comportement de formation de prix.

McFADZEAN D., (1995), "SimBioSys : A Class Framework for Evolutionary Simulations", *Master's Thesis*, Department of Computer Science, University of Calgary, Alberta, Canada, + Programme SimBioSys.

McFADZEAN D., TESFATSION L. (1999), "A C++ Platform for the Evolution of Trade Networks" , *Computational Economics*, N°14 , pp.109-134.

NAGURNEY A., (1996), "Parallel Computation", pp.335-406 in H.M.AMMAN, D.A.KENDRICK & J.RUST (EDS), *Handbook of Computational Economics*, Amsterdam, North-Holland.

NEWELL M.E., NEWELL R.G., T.L.SANCHA, (1972), "A solution to the hidden surface problem," in *Proceedings of the ACM National Conference*.

PETERSON G.L., (1981), "Myths about the Mutual Exclusion Problem", *Inf. Process. Lett.*, N°12-3, pp.115-16.

SCARF H., (1967), "The Approximation of Fixed Points of a Continuous Mapping", *SIAM Journal on Applied Mathematics*, Sept.

TANENBAUM A., (1996), *Computer Networks, Upper Saddle River*, Prentice Hall, (trad., Dunod, Paris, 1999).

TESFATSION L., (1997), "A Trade Network Game with Endogenous Partner Selection", pp.249-269., in H.AMMAN, B.RUSTEM, & A.B.WHINSTON (EDS),*Computational Approaches to Economic Problems*, Kluwer Academic Publishers.

VIENNOT X.G., (1999), "A Strahler Bijection between Planar Trees and Dyck Paths", *Actes du XIè Colloque Séries Formelles et Combinatoire Algébrique*, pp.573-84. Barcelone, juin.

ZADEH L.A., (1965), "Fuzzy Sets", *Information and Control*, N°8, pp.338-53.

ANNEXE 3.1 - LES MÉCANISMES DE MARCHÉ DU MODÈLE SINGUL

Chaque agent i rencontre N_i agents pour négocier une transaction[35]. Dans une première version du modèle - Fig.3.1.a, les agents contractent si les intervalles de prix sont compatibles, alors que dans la seconde - Fig.3.1.b - les agents classent les partenaires rencontrés et la transaction est effectuées avec le meilleur, si la réciproque est vraie.

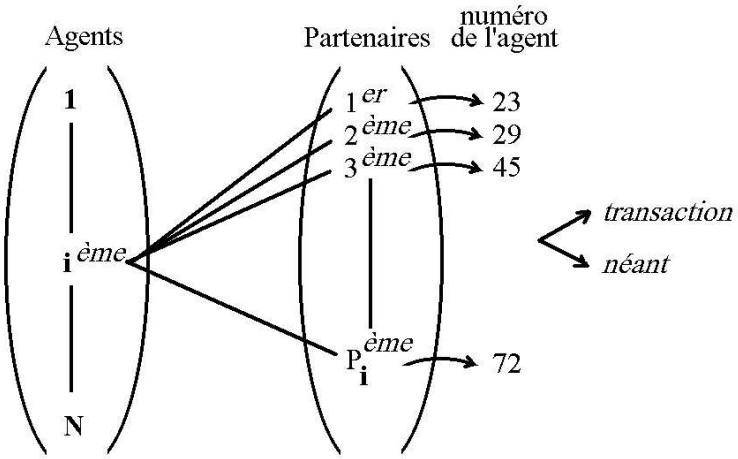


Fig.3.1.a - Mécanisme de recherche de partenaires sans classement dans le modèle SINGUL[36]

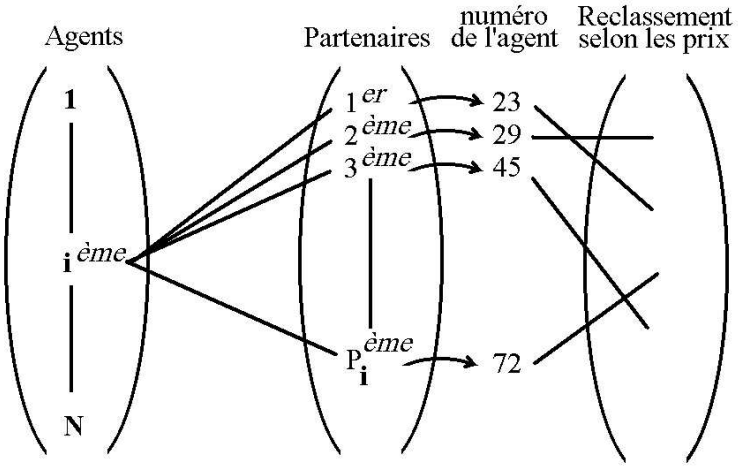


Fig.3.1.b - Mécanisme de recherche de partenaires avec classement dans le modèle SINGUL[37]

[35]- Dans une version exporatoire, Modèle d'Echanges et de Recherches Dynamiques d'Informations pour les Transactions (MEREDIT), les agents effectuaient leurs transactions avec la qualité des informations dont ils disposaient. L'objectif était de lever l'hypothèse de transparence - voir notre note (1994.c).

[36]- Dite procédure "*simonienne*".

[37]- Dite procédure "*stiglerienne*".

Dans le modèle, la transaction, lorsqu'elle a lieu, est fixée au prix $P_{i,j} = \alpha_{i,j}.P_i + (1 - \alpha_{i,j}).P_j$, où P_i est le prix souhaité par l'agent i et $\alpha_{i,j}$ l'échelle de désirabilité de la transaction des partenaires : $\alpha_{i,j} = \frac{\|\Delta S_i\|}{\|\Delta S_i\|+\|\Delta S_j\|}$ Plus la différence ΔS_i entre le stock désiré et le stock réel est grande, plus fort sera le désir de l'agent i de conclure l'affaire - voir les algorithmes Fig.3.2.b.

ANNEXE 3.2 - LES ALGORITHMES DE MARCHÉ DU MODÈLE SINGUL

```
i:=0;
Repéter
  i:=i+1;
  Initialisation
  Patrimoine;
  StockDésiré;
  StockCourant;
  PrixMinVendeur;
  Détermination du statut (Offeur / Vendeur);
  PrixMaxAcheteur;
  MargeNégociation;
Jusqu'à (i)=Effectif);

Temps:=0;
Repéter
  Temps:=Temps+1;
  i:=0;
  Repéter
    i:=i+1;
    Choix du rang des partenaires;
    j:=0;
    Répeter
      j:=j+1;
      Partenaire:=Partenaire[j];
      Test de compatibilité (Offreur vs Vendeur);
      Si (Compatibilité) Alors Négociation du prix;
      Si (Prix conforme à la marge) Alors Conclusion;
    jusqu'à (Partenaire[j]>Partainaire_max);
  Jusqu'à (i)=Effectif);
Jusqu'à (Temps)>=Horizon);
```

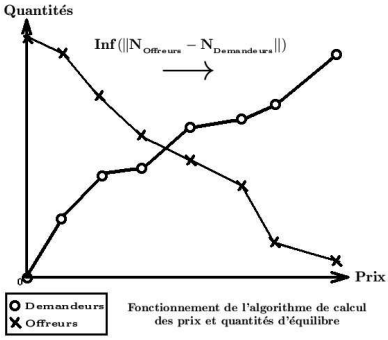
Version simonnienne du modèle SINGUL

```
i:=0;
Repéter
  i:=i+1;
  Initialisation
  Patrimoine;
  StockDésiré;
  StockCourant;
  PrixMinVendeur;
  Détermination du statut (Offeur / Vendeur);
  PrixMaxAcheteur;
  MargeNégociation;
Jusqu'à (i)=Effectif);

Temps:=0;
Repéter
  Temps:=Temps+1;
  Contruction de la matrice de contiguïté;
  i:=0;
  Repéter
    i:=i+1;
    j:=Partenaire_1
    Repéter
      Test de compatibilité (Offreur vs Vendeur);
      Si (Compatibilité) Alors Négociation du prix;
      Si (Prix conforme à la marge) Alors Réservation;
      j:=j+1;
    Jusqu'à (j)=Partenaire_Max);
    Classement des partenaires en fonction des prix;
  Jusqu'à (i)=Effectif);

  k:=0;
  Repéter
    k:=k+1;
    Si pour un couple d'agent (i,j) (rang(=k) Alors
      Debut
        Conclusion du contrat;
        Sortie de i et de j du marché;
      Fin;
  Jusqu'à (k)=Partenaire_max);
Jusqu'à (Temps)=Horizon);
```

Version stiglerienne du modèle SINGUL

Fig.3.2.a - Algorithmes de recherche de partenaires dans le modèle SINGUL

Par ailleurs, l'algorithme de détermination des prix et quantités d'équilibre ne fonctionne pas selon le critère de l'égalité stricte entre offre et demande. Autrement, l'algorithme risquerait de ne pas s'arrêter au point d'équilibre. En effet, dans le graphique de la Fig.3.2.b, les courbes sont obtenues par tracé entre les points observés et le point d'équilibre correspond à une "zone" non observée. C'est pourquoi l'algorithme itère sur les prix et dénombre la différence entre offreurs et demandeurs. Lorsque cette différence est minimale, alors on a atteint la zone d'équilibre.



Fonctionnement de l'algorithme de calcul des prix et quantités d'équilibre

```
Pas:=Pas0;
DiffMin:=1.0E+30;
Prix:=0.0;
i:=0;
Repéter
  i:=i+1;
  Prix:=Prix+PrixMin+Pas;
  Diff:=Abs(Offre[i]-Demande[i]);
  Si (Diff<DiffMin) Alors
    Début
      Rang:=i;
      PrixEqu:=Prix;
    Fin;
Jusqu'à (Prix>=PrixMax);
QuanEqu:=Inf(Offre[Rang],Demande[Rang]);
```

Algorithme de calcul de l'équilibre

Fig.3.2.b - Recherche des prix et quantités d'équilibre avec SINGUL

IV/ Précision des Calculs et Arithmétique des Ordinateurs

On ne peut présenter un panorama des algorithmes de la modélisation sans évoquer la question de la précision des calculs sur ordinateurs et plus généralement celle du codage des nombres par les ordinateurs. Cette question renvoie à l'Arithmétique, "*étude élémentaire des propriétés des nombres entiers et des nombres rationnels, établie avant le XVIIIè siecle* [...] *la Théorie de nombres, les développements nés des recherches précédentes à partir du XVIIIè*" (J.ITARD, *Arithmétique et Théorie des nombres*, Paris, PUF, Coll.Que sais-je ?, 1973, p.7). Même si l'Arithmétique se "limite" à l'étude des nombres, leur constitution en ensemble, les opérations que l'on peut leur appliquer, il ne faudrait pas en négliger la portée algorithmique. L'Arithmétique est à la base de la cryptologie (la science des codages), de la compression des données (qui réduit le coût de transmission téléphonique de données). Elle intervient en Astronomie pour des calculs de chronologie, enfin elle a permis de faire progresser la précision des calculs sur ordinateurs avec des propositions de reformulation complète des opérations.

a) LES PRINCIPALES APPLICATIONS DES ALGORITHMES DE L'ARITHMÉTIQUE

Divisibilité et primatialité - *i.e.* propriété des nombres qui ne sont divisibles que par 1 et par eux-mêmes - des nombres sont les deux propriétés essentiellement utilisées dans les applications de l'arithmétique. En particulier le codage, qui consiste à transformer un message intelligible dans un autre système de signes ou symboles (J.VÉLU, 1994, pp.269-304). La conversion de messages quelconques en une suite numérique pour la transmission téléphonique, la compression des données, constituent des codages. *a priori*, le codage n'est pas destiné à rendre le message inintelligible. On peut chercher une simplification, une correction, une détection et bien sûr une dissimulation (sûreté militaire, secrets financiers etc.). Comme nous le verrons, les techniques de codage travaillent beaucoup dans l'ensemble $\mathbb{Z}/m\mathbb{Z}$. - *i.e.* ensemble des restes de la division d'éléments de $\mathbb{Z}$ par m.

i - Généralités sur le codage

Changement de base - un des premiers codage simple consiste à changer de base de numération. On peut convertir un texte en nombre (par ex. chaque lettre est convertie selon son rang dans l'alphabet en $Base_{10}$, puis chaque nombre $Base_{10}$ est converti dans une autre base, $Base_c$). La conversion en p d'un entier n en base c s'obtient comme suit :

```
i :=0 ;
Repéter
   i :=i+1 ;
   a :=n mod c ;
   n :=n div c ;
   b[i] :=a ;
Jusqu'à (n=0) ;
```

Algorithme de changement de base

où *mod* est le reste de la division entière, *div*[1]. Si le nombre d'itérations i_{max} nécessaires pour la conversion de n en p, alors p s'obtiendra comme la suite de chiffres juxtaposés :

$$p = n\ b[i_{max}]\ b[i_{max} - 1]\ \ldots\ b[1]$$

Clé de sûreté - un autre codage consiste à accoler une clé à un nombre. Supposons un nombre N que nous noterons

$$z[n]\ z[n-1]\ ...\ z[1]$$

(où $z[1]$ correspond au chiffre des unités, $z[2]$ celui des dizaines etc. et n la taille de la mantisse nécessaire pour représenter ce nombre), on choisit la clé k en fonction d'un nombre p ayant une propriété arithmétique particulière, p est un nombre premier, et qui ne soit pas trop grand par rapport au nombre dont il sera la clé. k vérifie ainsi la relation

$$k = \left(\sum_{i=1}^{n} z[i] \right) \pmod{p}$$

où k est intrinsèquement liée à N et ce fait présente un intérêt dans l'optique du contrôle des numéros. Par exemple le numéro de compte bancaire doit être cohérent avec la clé (même si celle-ci est alphabétique). Mais on voit bien, dans notre exemple, que l'on peut commettre deux erreurs de saisie qui permettraient de retomber sur la bonne clé[2].

Les algorithmes de compression des données - il s'agit d'une forme de codage, qui comporte une propriété particulière par rapport à d'autres types de codage. La taille du message codé doit être significativement plus petite que celle du message originel (voir programmes et développements de M.NELSON, 1993). Dans la théorie de l'information, on définit la quantité d'informations I (en bits) d'un message comme

$$I = -Log_{10}(P)$$

où P est la probabilité d'occurrence du message. Les algorithmes de compression sont basés sur la redondance d'occurrence - *i.e.* le fait que telle lettre de l'alphabet revient plus régulièrement qu'une autre en Français. Concrètement, si un message comporte plusieurs fois de suite le même signe, il est plus économique d'écrire une seule fois le signe en précisant combien de fois il apparaît.

L'Algorithme de Shannon-Fano (C.E.SHANNON, 1951 et R.M.FANO, 1961) consiste, à partir de la probabilité d'occurrence de chaque symbole dans un message, à construire une table de code dont la taille (en bits) est inversement proportionnelle à la probabilité d'occurrence. Prenons l'exemple de M.NELSON (*op.cit.*, pp.24-25). On dispose de symboles que l'on a dénombrés dans le message ; ils sont classés par ordre décroissant d'occurrences - Étape1.

[1]- Voir P.GOETGHELUCK, *Cours d'arithmétique*, Mimeo, Université d'Orsay Paris-Sud, 1997, 35 p.

[2]- Ce n'est pas un hasard si des listes de numéros de série de logiciels circulent frauduleusement sur Internet.

Symbole	Compte
A	15
B	7
C	6
D	6
E	5

Algorithme de Shannon-Fano - Étape 1

On divise le tableau en deux groupes à la médiane et on affecte le chiffre 0 au dessus du tableau et le chiffre 1 au dessous du tableau pour les symboles correspondants - Étape 2.

Symbole	Compte	Division(s)
A	15	0
B	7	0
--------- $1^e div$		
C	6	1
D	6	1
E	5	1

Algorithme de Shannon-Fano - Étape 2

On réitère en divisant chaque sous-groupe en deux, en affectant de nouveau les valeurs 0 et 1 en fonction de la position du sous-groupe - Étape 3 - et chaque symbole se trouve codé.

Symbole	Compte	Division(s)
A	15	0 0
---------- $2^e div$		
B	7	0 1
-------- $1^e div$		
C	6	1 0
---------- $3^e div$		
D	6	1 1 0
------------ $4^e div$		
E	5	1 1 1

Algorithme de Shannon-Fano - Étape 3

Citons, l'Algorithme de Huffman (D.A.HUFFMAN, 1952) qui est similaire au précédant quoique, entre autres différences, les itérations se déroulent en sens inverse. L'Algorithme arithmétique consiste quant à lui, à coder les symboles non plus en binaire, mais au moyen d'un nombre en virgule flottante (codage arithmétique des nombres sur l'ordinateur). Enfin J.ZIV et A.LEMPEL (1977-78) ont proposé un Algorithme à base de dictionnaire qui code chaque occurrence trouvée (mot) selon sa position dans un dictionnaire conventionnel (physique ou logiciel) - ils sont à la base des programmes actuels de compression.

ii - La cryptologie

Ce type de codage recourt à des techniques de substitution de signes et/ou d'alphabets, des transpositions avec la fonction *modulo* (calcul dans $\mathbb{Z}/m\mathbb{Z}$). Les applications de $\mathbb{Z} \rightarrow \mathbb{Z}/m\mathbb{Z}$ ne sont pas bijectives, *i.e.* un même message codé peut correspondre à une infinité de messages potentiels tant que l'on ne dispose pas de la clé de décryptage. On trouve également des cryptages basés sur le calcul matriciel (B.BECKETT, 1990, pp.193-201).

La cryptographie RSA à clé publique - la technique de codage la plus connue est sans doute celle de l'Algorithme RSA (R.RIVEST, A.SHAMIR et L.ADLEMAN, 1977). Voyons concrètement comment cet algorithme est mis en oeuvre. Deux personnes doivent communiquer entre elles, S qui envoie le message et R qui le reçoit. Lors de la phase de codage, R choisit deux entiers p et q premiers entre eux - *i.e.* n'ayant pas de diviseur commun - et à 200 chiffres environ. Il calcule $n = p.q$ et $\phi(n) = (p-1).(q-1)$ puis choisit la clé de cryptage

$$e \ / \ PGCD\{e, \phi(n)\} = 1$$

Les valeurs e et n sont communiquées publiquement, mais p, q et $\phi(n)$ sont gardées secrètes. Le message doit être codé en une suite d'entiers a $a < n$ et a et n premiers entre eux. S envoie alors à M le résultat de l'équation suivante :

$$c = a^e \pmod{n}$$

Lors de la phase de décodage, R calcule la clé de décryptage en résolvant l'équation en d suivante :

$$e.d \ \equiv \ 1 \ mod \ \phi(n)$$

avec $1 < d < \phi(n)$. Grâce au Théorème d'Euler, appelé aussi "*petit théorème de Fermat*" (P.GOETGHELUCK, *op.cit.*, pp.27-28), on obtient la simplification : $c^d = a \pmod{n}$.

Exemple

1°- Phase de codage, $n = 33$ d'où $p = 2$ et $q = 11$ Il vient que $\phi(n) = (2 - 1).(11 - 1) = 20$. Si l'on choisit $a = 13$, le cryptage donne $c = 13^3 \ mod \ 33 = 19$.

2°- Phase de décodage, on calcule $3.d \ \equiv \ 1 \ mod \ 20$ d'où $d = 7$.

iii - La précision calendaire

Dans notre calendrier, la plupart des calculs de datation se fondent sur une date de référence (la plus ancienne possible) et sur la manipulation de la fonction modulo - *i.e.* calculs dans $\mathbb{Z}/m\mathbb{Z}$ avec $m = 7$ (P.GOETGHELUCK, *op.cit.*). Nous ne ferons ici ni la présentation de tous les calendriers (lunaires ou solaires) ni l'historique de notre calendrier - *i.e.* le calendrier grégorien adopté en 1582 par l'Italie puis progressivement par tous les pays industrialisés jusqu'à devenir mondial. Si nous abordons ce thème, c'est qu'il est devenu indispensable au modélisateur de tenir compte de l'hétérogénéité des trimestres, des mois et même des jours de l'année[3]. Il s'agit pour le modélisateur d'effectuer la Correction

[3]- Un problème mis en évidence par S.C.HILLMER (1982) et W.R.BELL & S.C.HILLMER (1983).

des Variations Saisonnières (CVS) ainsi que la Correction des Jours Ouvrables (CJO) dans les projections conjoncturelles. Ainsi, le problème consiste à calculer le nombre de jours écoulés sur une période donnée et le rang de tel ou tel jour particulier (férié) dans l'année. Dans le TABLEAU N°4, on voit nettement que l'omission des jours fériés constitue une erreur non négligeable compte tenu de leur nombre et de leur caractère systématique[4] ; en effet ceux-ci sont prévisibles par les agents économiques au début de chaque année (sur simple consultation de leur agenda ou calendrier) et donc intégrables dans leurs décisions.

TABLEAU N°4 - Omission des fêtes légales[5]

Décennies	0	1	2	3	4	5	6	7	8	9
1900	9	10	9	9	8	9	9	10	9	7
1910	7	9	13	9	9	7	9	9	10	10
1920	7	7	9	9	10	9	7	7	8	10
1930	10	9	8	9	9	10	9	7	7	9
1940	13	10	9	7	9	9	10	10	7	7
1950	9	9	10	9	7	7	8	10	10	9
1960	8	9	9	10	9	7	7	9	13	10
1970	9	7	9	9	10	9	7	7	9	9
1980	10	9	7	7	8	10	9	9	8	9
1990	9	10	9	7	7	9	13	9	9	7
TOTAL : 881 JOURS										

Les algorithmes de périodisation quotidienne - Le traitement de la périodicité quotidienne implique la prise en considération de la séquence des jours de la semaine dans l'ordre (Dimanche, Lundi....) ainsi que le rang des jours fériés dans l'année[6]. Si l'on raisonne pour un pays donné[7], il existe deux types de fêtes légales : 1°- les fêtes fixes (le $i$ème jour du $j$ème mois) et les fêtes mobiles (par ex., le $k$ème dimanche du $j$ème mois). Deux calculs conventionnels seront nécessaires pour déterminer ces paramètres calendaires : le calcul de la bissextilité de l'année et le calcul de la date de Pâques. La bissextilité permet de déterminer si l'année compte 365 ou 366 jours dans l'année, alors que la fête de Pâques permet de déterminer la plupart des fêtes mobiles religieuses par simple translation (J.MÉEUS, 1986).

[4]- Cela étant, en économétrie, l'effet de calendrier n'apparaît pas systématiquement à tous les niveaux d'agrégation temporelle (J.BOSREDON et al., 1999).

[5]- Ce tableau a été obtenu grâce au module CHRONO que nous avons construit pour le système SIMUL. Il a été conçu à partir d'un premier logiciel de calendrier grégorien (GRECAL). CHRONO est un module de calendrier grégorien itératif qui génère les fêtes quotidiennes françaises, ainsi que des vecteurs indicateurs mensuels, trimestriels du nombre de jours fériés ou ouvrables de l'année souhaitée. Il peut également extrapoler arbitrairement les ponts - à ce propos voir nos notes 1996.b et 1997.a. A propos de notre logiciel GRECAL (initialement en FORTRAN) présenté à la Société Astronomique de France, voir H.ROY, "Compte rendu de la réunion Astronomie & Informatique du 8 octobre 1988", *L'Astronomie*, fév., 1989, pp.91-92.

[6]- Il ne s'agit pas seulement de tradition ; les dates de ces fêtes sont inscrites au journal officiel - voir à ce propos BUREAU DES LONGITUDES, *Éphémérides*, Paris, Gautiers-Villars, 1990, pp.7-21.

[7]- Qu'ils observent ou non le calendrier grégorien, tous les pays ne disposent pas des mêmes fêtes.

```
IF (AN MOD 4<>0) THEN
  BEGIN
    NJOUR :=365 ;
END
ELSE
  BEGIN
    IF ((AN MOD 100=0)
    AND (AN MOD 400<>0))
      THEN NJOUR :=365 ;
      ELSE NJOUR :=366 ;
  END ;
```

Fig.33 - Algorithme de bissextilité du millésime

La bissextilité de l'année - La terre n'effectue pas exactement sa rotation autour du soleil en 365 jours, il faut faire un rattrapage - *i.e.* un jour supplémentaire dans l'année toutes les p années. Le calendrier julien effectuait ce rattrapage tous les 100 ans, mais cela était insuffisant, c'est pourquoi le Pape Grégoire XIII décida d'instaurer avec le calendrier grégorien, un rattrapage tous les 4 ans et une année séculaire sur 4. Ainsi, l'année courante AN sera commune $NJOUR = 365$ si AN n'est pas divisible par 4. Si AN est divisible par 4, par 100 mais pas par 400 alors l'année est commune. Si AN est divisible par 4, par 100 et par 400 alors l'année est bissextile $NJOUR = 366$ - voir Fig.33.

```
BA := AN MOD 19 ;
BB := AN MOD 100 ;
B := AN DIV 100 ;
E := B MOD 4 ;
D := B DIV 4 ;
Z := B+8 ;
F := Z DIV 25 ;
Y := B-F+1 ;
G := Y DIV 3 ;
W := 19*BA+B+15-(D+G) ;
IH := W MOD 30 ;
IK := BB MOD 4 ;
I := BB DIV 4 ;
V := 2*(E+I)+32-(IH+IK) ;
Q := V MOD 7 ;
U := 11*(IH+2*Q)+BA ;
M := U DIV 451 ;
BT := 114+IH+Q-7*M ;
P := BT MOD 31 ;
N := BT DIV 31 ;
IJOUR := P+1 ;
MOIS :=N ;
```

Fig.34- Algorithme de calcul de Pâques de Spencer-jones

La fête de Pâques - Conventionnellement, Pâques est une fête chrétienne fixée en fonction du rythme de la Lune (au premier Dimanche suivant la première pleine lune de l'équinoxe de Printemps). Il existe deux algorithmes de calcul de la date de Pâques. Celui de Gauss, qui est valable uniquement sur la période grégorienne (1582 et au delà) et celui de Spencer-Jones (*General Astronomy*, London, 1961) valable sur les périodes julienne et grégorienne[8]. Nous ne donnons ici que le second, qui est couramment employé en astronomie - voir la Fig.34 où

[8]- Voir J.MÉEUS (1986, *op.cit.*, p.23).

AN est l'année courante, $IJOUR$ le jour de la semaine (0=Dimanche, 1=Lundi etc.) et $MOIS$ comme son nom l'indique est le mois[9].

Le traitement des saisonnalités - La répétition de séquences temporelles successives (quotidiennes, mensuelles, trimestrielles etc.) peut être intégrée dans des modèles. Il peut en effet être nécessaire de corriger les effets saisonniers qui masquent l'évolution moyenne. Il existe de nombreuses méthodes[10] dont le principe général consiste à isoler la composante saisonnière d'une série chronologique soit par des procédés économétriques (on estime les coefficients saisonniers), soit par moyennes mobiles (on rapporte les moyennes mensuelles à la moyenne annuelle)[11].

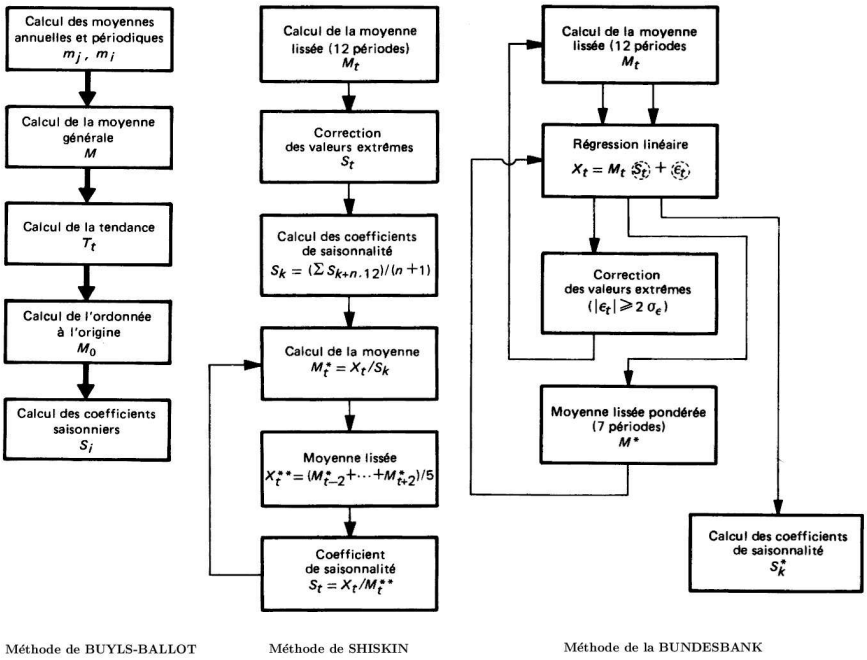


Fig.35 - Trois méthodes de désaisonnalisation[12]

[9]- A propos des problèmes de rattrapage de la course de la terre autour du soleil par le calendrier grégorien et de la recherche de stabilisation de la date de Pâques dans le calendrier, voir J.P.PARISOT et F.SUAGHER (*Calendriers et chronologie*, Paris, Masson, Coll. De Caelo, pp.81-84).

[10]- H.MENDERHAUSEN ("Methods of Computing and Eliminating Changing Seasonal Fluctuations", *Econometrica*, 1937) en donne une étude comparative.

[11]- Les logiciels les plus utilisés par les conjoncturistes sont DEMETRA, CENSUS X11 et TRAMO.A propos d'un historique des méthodes et logiciels, voir K.ATTAL (1996).

[12]- D'après R.LEVANDOWSKI (*La prévision à court terme*, Paris, Dunod, 1979, pp.141-69).

b) LE CONTRÔLE DE LA PRÉCISION DES CALCULS SUR ORDINATEURS

Lorsque le premier ordinateur - l'ENIAC - sortit les tout premiers résultats numériques, on raconte que l'un de ses promoteurs, John Von NEUMANN, fut très déçu. La représentation imparfaite des nombres par les ordinateurs se traduit en effet par des risques non négligeables d'erreurs de calcul. Dans un premier temps nous poserons le problème, puis nous exposerons les solutions proposées par les spécialistes, ainsi que notre propre outil, GNOMBR, développé pour la modélisation. En marge du problème de précision, nous évoquerons enfin le problème ergonomique de la lisibilité des mantisses.

i - L'énoncé du problème

Non seulement les PC de bureau sont vulnérables, mais les supercalculateurs aussi (exemple 1), et le recourt au calcul formel n'y change rien (exemple 2) - voir également les erreurs sur la fonction factorielle en Annexe 4.1.

Exemple N°1 - Le système[13]

$$\begin{cases} r = 9x^4 - y^4 + 2y^3 \\ x = 10864 \\ y = 18817 \end{cases}$$

donnent les résultats suivants :

Machines	Résultats
CDC-7600	0
CRAY ONE	2
VAX	7.081589E+08

alors que le résultat exacte est $r = 1$.

Exemple N°2 - La suite[14]

$$\begin{cases} u_0 = 2 \\ u_1 = -4 \\ u_{n+1} = 111 - \frac{1130}{u_n} + \frac{3000}{u_n . u_{n-1}} \end{cases}$$

donne $u_{90} = 99.998040316541816165\ldots$ avec le logiciel de calcul formel Maple, alors que le résultat exacte est $u_{90} = 6.0000000996842706405\ldots$

La norme **IEEE-754** dite arithmétique en virgule flottante - la représentation des nombres réels par les ordinateurs, dite en en virgule flottante (**v.f.**), est très imparfaite en termes de bornes, mais également en termes de continuité. Il est ainsi en réalité tout à fait illusoire de parler de calculs de limite, de supposer

[13]- D'après M.PICHAT & J.VIGNES (1993, pp.3-5).
[14]- D'après M.DAUMAS et J.M.MULLER (EDS.) (1997, pp.2-8).

la continuité des fonctions et donc, *a fortiori*, de proposer de les dériver[15]. Tout élément X de l'ensemble $\mathbb{R}$ des réels est représenté dans l'ensemble $\mathbb{F}$ des réels "flottants" comme suit[16] : $X = e.M.b^E$ où e est le signe de X, M la mantisse codée sur t chiffres de la base b, et E l'exposant. L'ordinateur opère une troncature des nombres qui se traduit par le fait que le dernier chiffre de la mantisse est arrondi[17].

Il existe donc deux problèmes :

1°- la machine est incapable d'atteindre les limites infinies de l'ensemble des réels ou des entiers (Ex. : `1.0E+40` qui est la limite de représention des réels simples, est un grand nombre, mais peut-on parler d'infini ?[18]).

2°- la machine ne reproduit pas non plus la propriété de séparabilité de l'ensemble réels. (Ex :`1.0E+40 + 1 = 1.0E+40`, lorsque deux opérandes d'une addition n'ont pas le même ordre de grandeur, le plus petit des deux disparaît dans l'approximation de l'opération). Ajoutons que, l'on ne peut malheureusement affirmer que le nombre réel exact est minoré ou majoré par son représentant en v.f. Il peut y avoir arrondi par excès ou par défaut (`E.FFFFFF47` donne `E.FFFFFF5`, mais `E.FFFFFF44` donne également `E.FFFFFF4`)[19].

Cependant, tous les algorihmes ne présentent pas la même vulnérabilité vis à vis de la **v.f.** ; M.PICHAT et J.VIGNES (*op.cit.*) en ont proposé une classification :

1°- Les algorithmes finis - algorithmes qui proposent une solution exacte lorsqu'ils sont calculés en arithmétique exacte. Les algorithmes de calcul matriciel usuels appartiennent à cette catégorie (élimination de Gauss...) ;

2°- Les algorithmes itératifs - dont la solution est définie à partir d'un intervalle de convergence. Les algorithmes de résolution tels que Gauss-Seidel, Newton... Dans ce cas l'algorithme est convergeant si $F(X_q) = \underline{0}$; stationnaire si $|X_q - X_{q-1}| = 0$; divergent si $q > N_{n_{max}}$ - où F représente l'algorithme de résolution sur des variables X appliqué au cours de q itérations et $\underline{0}$ représente le zéro informatique qui est significativement proche de sa valeur mathématique.

---

[15]- Tout au moins, au voisinage des nombres irrationnels.

[16]- Voir M.LA PORTE & J.VIGNES (1974, *Algorithmiques numériques - Tome 1*, Paris, Technip, 226 p.), notamment pour une formalisation mathématique plus complète (*Ibid.*, pp.17-39).

[17]- Pour des compléments au sujet des problèmes d'arithmétique flottante voir J.BERSTEL et al. (1991, *op.cit.*, pp.123-204). Voir les programmes Pascal et l'argumentation de N.WIRTH (*op.cit.*, pp.94-100) qui préconise d'adopter une représentation binaire et non pas la représentation décimale qui nous est familière. Il appuie son propos en appliquant sa méthode avec les chiffres romains.

[18]- Une nouvelle arithmétique a défini la notion de grands nombres et de petits nombres. Selon l'Analyse Non standard dite "théorie des ordres de grandeurs" de A.ROBINSON (1960), $\mathbb{R}^+$ est divisé en trois classes de nombres : les nombres idéalement petits (i-petits), les nombres appréciables et les nombres idéalement grands (i-grands) - voir à ce propos M.DIENER et A. DELEDICQ, *Leçon de calcul infinitésimal*, Paris, A.Colin, 1989.

[19]- MAILLÉ M. (1982, "Some Methods to Estimate Accuracy of Measurements or Numerical Computations", *Processing of Mathematics for Computer*, Congress AFCET) a cependant montré que le nombre de chiffres significatifs de la **v.f.** suivait une loi Laplace-Gauss.

3°- Les algorithmes approchés - algorithmes de calcul différentiel qui induit une erreur due au pas de discrétisation (des intégrales, dérivées etc).

ii - Les pistes de contrôle algébrique

Ces techniques reposent sur la présentation des calculs, *i.e.* elles font en sorte d'éviter des calculs intermédiaires entraînant des arrondis catastrophiques (Conditionnement) et elles suppriment les redondances, notamment dans les polynômes (Méthode de Horner).

Le conditionnement d'un système consiste à calculer

$$Cond(X) = \frac{\sqrt{\lambda_{max}}}{\sqrt{\lambda_{min}}}$$

pour savoir si le facteur de perturbation qui transforme

$$X.a = b$$

en

$$(X + \Delta.X).a = (b + \Delta.b)$$

est significatif[20]. Si $Cond(X)$ 1 alors la matrice est bien conditionnée, sinon on change de pivot.

L'Algorithme de Horner transforme le polynôme

$$P(x) = \sum_{i=1}^{n} a_i x^i$$

comme suit :

$$P(x) = a_n.x^n + a_{n-1}.x^{n-1} + \ldots + a_1.x^1 + a_0.x^0$$

$$\Rightarrow \; P(x) = x(x(\ldots x(a_n x + a_{n-1}) + \ldots)) + a_1) + a_0$$

```
FUNCTION EVAL( P :POLYNOME; X :REAL; N :INTEGER) : REAL;
VAR i :INTEGER;
  r :REAL;
BEGIN
  r :=P[n];
  FOR i :=n-1 DOWNTO 0 DO EVAL :=r*P[i];
END;
```

Fig.36 - Algorithme de Horner[21]

iii - Les pistes de contrôle arithmétique

Plusieurs types de recherches ont été menées : la première catégorie de travaux a conservé peu ou prou la base de numération et la logique des opérateurs

[20]- Les λ_i étants les valeurs propres de la matrice X.
[21]- D'après P.HERNERT (1995, *op.cit.*).

(la base 10 et les opérations arithmétiques de base). On trouve ainsi des travaux dont l'objet est de déterminer le nombre de chiffres significatifs pour un processus donné[22], et ceux dont l'arithmétique est plus précise[23]. La seconde catégorie change totalement de numération voire d'opérations.

Les arithmétiques stochastiques : la méthode CESTAC - Cette technique consiste à estimer l'erreur de précision commise sur un résultat de calcul, en évaluant les paramètres de la distribution de probabilités d'erreur. Mais dans la mesure où la partie perdue lors des arrondis n'est par définition pas observable, les procédures mises au point consistent à simuler une perturbation (pour plus de développements, voir M.PICHAT et J.VIGNES, *op.cit.*, pp.89-189 ; ainsi que M.DAUMAS et J.M.MULLER (EDS.), *op.cit.*, pp.65-92).

L'arithmétique d'intervalles (G.ALEFELD et J.HERZBERGER, 1983) - moyennant des vérifications quant au domaine de validité de cette arithmétique[24] il a été possible de proposer des algorithmes élémentaires effectuant des calculs non plus sur des nombres a mais sur son intervalle de précision : $[a] \rightarrow [\underline{a}, \overline{a}]$. Ainsi, les opérations élémentaires ont été définies de la manière suivante :

TABLEAU N°5 - Algorithmes d'intervalles

`Addition`	$[a] + [b]$	$[\underline{a} + \underline{b}, \overline{a} + \overline{b}]$
`Soustraction`	$[a] - [b]$	$[\underline{a} - \underline{b}, \overline{a} - \overline{b}]$
`Multiplication`	$[a][b]$	$[Min(\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{ab}), Max(\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{ab})]$
`Division`	$[a]/[b]$	$[\underline{a}, \overline{a}][\frac{1}{b}, \frac{1}{\underline{b}}]$ `si` $0 \notin [b]$

Les Algorithmes en multiprécision, *i.e.* si le nombre de chiffres est significativement grand mais, au delà de cette précision il peut y avoir un arrondi, sont présentés par D.E.KNUTH (*op.cit.*, pp.250-65), J.BERSTEL et al. (*op.cit.*, 1991, tome 2, pp.149-204) ainsi que M.DAUMAS et JM.MULLER (EDS.) (*op.cit.*, pp.93-116). Il s'agit en fait de reprogrammer les opérations arithmétiques élémentaires pour qu'elles gèrent un grand nombre de chiffres. C'est ainsi que nous avons procédé pour programmer notre module GNOMBR (voir nos notes de travail 1996.c et 1998). Nous avons implémenté les algorithmes arithmétiques "scolaires"[25] (addition, soustraction etc.) en tenant compte des règles de signes et en vérifiant l'exactitude des résultats grâce à des procédures du type "preuve par neuf" (voir à ce propos P.GOETGHELUCK, *op.cit.*). La plupart des arithmétiques en multiprécision fournissent des bibliothèques permettant aux programmeurs de recourir à celles-ci de manière souple (GrandNombres en Turbo-

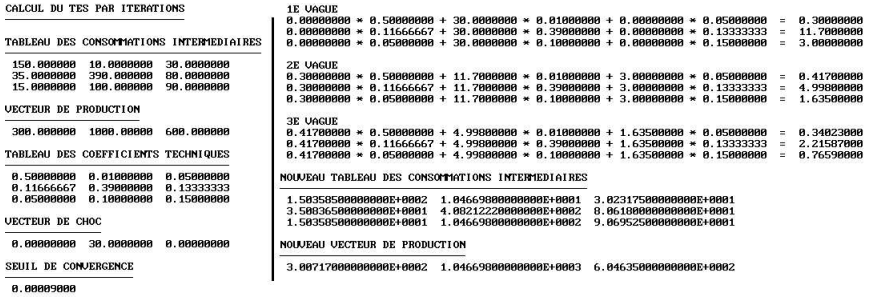
[22]- La méthode CESTAC (**C**ontrôle et **E**stimation **ST**ochatique des **A**rrondis de **C**alculs), M.PICHAT & J.VIGNES (1993, pp.170-175) proposent d'estimer la propagation des erreurs d'arrondis pour déterminer le nombre de chiffres significatifs.

[23]- U.KULISCH et W.N.MIRANKER (*Computer Arithmetic in Theory and Practice*, New York, Academic Press, 1981), ont proposé une méthode déterministe, basée sur la construction d'une arithmétique exacte pour la seule opération de produit scalaire, permettant de contrôler les erreurs sur les autres opérations.

[24]- Voir à ce propos M.DAUMAS et J.M.MULLER (EDS) (*op.cit.*, pp.41-64).

[25]- Voir en Annexe 4.4.

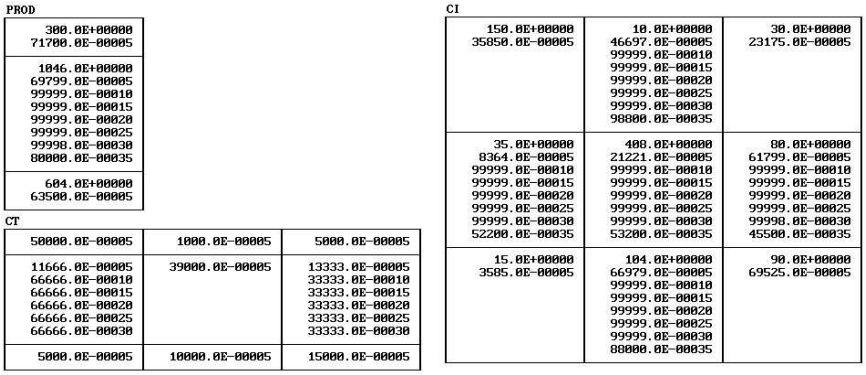
Pascal dans le cas de J.BERSTEL et al., 1991 ; M.DAUMAS et J.M.MULLER (EDS) citent MP en FORTRAN, MP GNU en C, MP FUN en C, PARI en C/C++ et RANGE en C++). En faisant le logiciel GNOMBR[26], notre objectif était d'étudier les différences de précision que l'on pouvait attendre avec un même modèle en virgule flottante et en multiprécision. A titre d'application, nous avons converti un modèle TES itératif simplifié (ITERTES), en un modèle multiprécision (GN_TES) - voir en Annexe 4.2 le programme ITERTES avec les instructions permettant sa conversion par le programme EDIT_GN.



```
CALCUL DU TES PAR ITERATIONS

TABLEAU DES CONSOMMATIONS INTERMEDIAIRES
 150.000000  10.0000000  30.0000000
 35.0000000  390.000000  80.0000000
 15.0000000  100.000000  90.0000000
VECTEUR DE PRODUCTION
 300.000000  1000.00000  600.000000
TABLEAU DES COEFFICIENTS TECHNIQUES
 0.50000000  0.01000000  0.05000000
 0.11666667  0.39000000  0.13333333
 0.05000000  0.10000000  0.15000000
VECTEUR DE CHOC
 0.00000000  30.0000000  0.00000000
SEUIL DE CONVERGENCE
 0.00009000

1E VAGUE
0.00000000 * 0.50000000 + 30.0000000 * 0.01000000 + 0.00000000 * 0.05000000  =  0.30000000
0.00000000 * 0.11666667 + 30.0000000 * 0.39000000 + 0.00000000 * 0.13333333  =  11.7000000
0.00000000 * 0.05000000 + 30.0000000 * 0.10000000 + 0.00000000 * 0.15000000  =  3.00000000

2E VAGUE
0.30000000 * 0.50000000 + 11.7000000 * 0.01000000 + 3.00000000 * 0.05000000  =  0.41700000
0.30000000 * 0.11666667 + 11.7000000 * 0.39000000 + 3.00000000 * 0.13333333  =  4.99800000
0.30000000 * 0.05000000 + 11.7000000 * 0.10000000 + 3.00000000 * 0.15000000  =  1.63500000

3E VAGUE
0.41700000 * 0.50000000 + 4.99800000 * 0.01000000 + 1.63500000 * 0.05000000  =  0.34023000
0.41700000 * 0.11666667 + 4.99800000 * 0.39000000 + 1.63500000 * 0.13333333  =  2.21587000
0.41700000 * 0.05000000 + 4.99800000 * 0.10000000 + 1.63500000 * 0.15000000  =  0.76590000

NOUVEAU TABLEAU DES CONSOMMATIONS INTERMEDIAIRES
 1.50358580000000E+0002  1.04669080000000E+0001  3.02317500000000E+0001
 3.50836500000000E+0001  4.00212220000000E+0002  8.06180000000000E+0001
 1.50358580000000E+0001  1.04669080000000E+0002  9.06952500000000E+0001
NOUVEAU VECTEUR DE PRODUCTION
 3.00717000000000E+0002  1.04669080000000E+0003  6.04635000000000E+0002
```

Fig.37 - Listing des résultats du programme ITERTES en virgule flottante

La comparaison des résultats - Fig.37 et 38 - fait apparaître les différences liées au mode de calcul. Bien entendu, compte tenu de la précision des données fictives fournies pour cette simulation, les différences observées ne sont pas significatives, mais cet exemple constitue un encouragement pour continuer dans cette voie.



PROD

300.0E+00000
71700.0E-00005
1046.0E+00000
69799.0E-00005
99999.0E-00010
99999.0E-00015
99999.0E-00020
99999.0E-00025
99998.0E-00030
80000.0E-00035
604.0E+00000
63500.0E-00005

CT

50000.0E-00005	1000.0E-00005	5000.0E-00005
11666.0E-00005	39000.0E-00005	13333.0E-00005
66666.0E-00010		33333.0E-00010
66666.0E-00015		33333.0E-00015
66666.0E-00020		33333.0E-00020
66666.0E-00025		33333.0E-00025
66666.0E-00030		33333.0E-00030
5000.0E-00005	10000.0E-00005	15000.0E-00005

CI

150.0E+00000	10.0E+00000	30.0E+00000
35850.0E-00005	46697.0E-00005	23175.0E-00005
	99999.0E-00010	
	99999.0E-00015	
	99999.0E-00020	
	99999.0E-00025	
	99999.0E-00030	
	98800.0E-00035	
35.0E+00000	400.0E+00000	80.0E+00000
8364.0E-00005	21221.0E-00005	61799.0E-00005
99999.0E-00010	99999.0E-00010	99999.0E-00010
99999.0E-00015	99999.0E-00015	99999.0E-00015
99999.0E-00020	99999.0E-00020	99999.0E-00020
99999.0E-00025	99999.0E-00025	99999.0E-00025
99999.0E-00030	99999.0E-00030	99998.0E-00030
52200.0E-00035	53200.0E-00035	45500.0E-00035
15.0E+00000	104.0E+00000	90.0E+00000
3585.0E-00005	66979.0E-00005	69525.0E-00005
	99999.0E-00010	
	99999.0E-00015	
	99999.0E-00020	
	99999.0E-00025	
	99999.0E-00030	
	88000.0E-00035	

Fig.38 - Listing des résultats du programme de GN_TES en multiprécision

L'arithmétique exacte s'applique aux algorithmes qui fournissent tous les chiffres sans erreur ni arrondi. Il faut préciser ici qu'il ne sont opérationnels qu'en présence de nombres entiers ou de nombres rationnels - il est donc hors de

[26]- Le logiciel GNOMBR est accompagné des programmes EDIT_GN (qui convertit un programme écrit en Turbo-Pascal pour qu'il intègre la multi-précision) et de TABL_GN (qui convertit des tableaux de résultats en multiprécision pour qu'ils soient plus facilement lisibles) - la Fig.37 est obtenue avec TABL_GN.

question d'extraire des racines carrées. Citons l'Arithmétique rationnelle paresseuse qui consiste à éviter de calculer les formes rationnelles, mais de n'utiliser que l'intervalle où elles sont définies ; le calcul n'a lieu - et encore ne fait-il intervenir que les intervalles de nombres situés à proximité de celui calculé - que lorsqu'il y a un doute sur l'intervalle dans lequel est censé se situer le nombre - voir M.DAUMAS et J.M.MULLER (EDS) (*op.cit.*, pp.126-29).

L'Arithmétique algébrique utilise le Théorème de Bezout (si A et B premiers entre eux, $\exists\ U, V\ /\ U.A + V.B = 1$). Si l'on pose la relation polynômiale

$$A(X) = Q(X).\phi(X) + R(X)$$

ainsi que $a = A(\alpha)$ et $b = B(\alpha)$ alors on définit les opérations arithmétiques comme suit :

TABLEAU N°6 - Algorithmes d'intervalles

`Addition`	$a + b$	$A(\alpha) + B(\alpha) = (A + B)(\alpha)$
`Soustraction`	$a - b$	$A(\alpha) - B(\alpha) = (A - B)(\alpha)$
`Multiplication`	ab	$A(\alpha).B(\alpha) = (AB\ mod\ \phi)(\alpha)$
`Inverse`	$1/a$	$U(\alpha)$

Les arithmétiques dynamiques - on doit présenter ici l'Algorithme d'Avizienis (A.AVIZIENIS, 1961), qui supposant une base b quelconque, utilise non pas les chiffres $\{1, \ldots\ b-1\}$ mais $\{-a, \ldots\ +a\}$ avec $a\ /\ a < b - 1$.

Exemple : $(1745)_{10}$[27] s'écrit $[2][-3][4][5]$ ou bien encore $[2][-3][4][-5]$. Si $a\ /\ \{2a \geq b - 1\ et\ a \leq b - 1\}$ on obtient que le calcul des retenues est indépendant des autres calculs, si bien que l'on effectue les calculs de gauche à droite (en sens inverse de l'habitude). Il existe plusieurs variantes de cette arithmétique - voir M.DAUMAS & J.M.MULLER (EDS), *op.cit.*, pp.142-50 ; à propos des propositions d'opérations arithmétiques, J.M.MULLER (1989, pp.55-165). L'amélioration de la précision se fait au détriment de la place mémoire et/ou de la rapidité des calculs[28].

iv - Problème ergonomique de lisibilité des mantisses de résultats

L'ergonomie informatique dans un logiciel de modélisation multidimensionnelle est impérative[29] car il s'agit de lire une très grande quantité de données numériques (résultats, coefficients, statistiques etc.) dont les ordres de grandeurs peuvent être très divers[30].

[27]- Lire `1745` en base `10`.

[28]- Voir à ce propos V.MÉNISSIER, *Arithmétique exacte : conception, algorithmique et performances d'une implantation informatique en précision arbitraire*, Thèse de Doctorat, Université de Paris VI, 1994.

[29]- Voir à ce propos notre note (1995.b) ainsi que le Chap.3 de notre Thèse de Doctorat (en cours).

[30]- Difficile de vite reconnaître dans `2.45E-0003` un taux de mortalité ou dans `1.235E+0004` un effectif.

TABLEAU N°7 - Comparaison de mantisses de résultats

Initiale	Forcée (:8)	Forcée (:25.8)	Avec MANTISSE
-.00000000000006	-6.0E-14	-0.00000000	-6.00E-0014
2.45E-0003	2.5E-03	0.00245000	0.00245000
1.235E+0004	1.2E+04	12350.00000000	12350.0000
-12453464367.13	-1.2E+10	-12453464367.00000000	-1.25E+0010
34646.	3.5E+04	34646.00000000	34646.0000

Nous avons donc implémenté dans le logiciel SIMUL une procédure de mantisse "intelligible" paramétrable - voir le résultat de l'affichage dans le TABLEAU N°7 et la procédure MANTIS en Annexe 4.3. Bien entendu, étant donné la perte d'information qu'elle peut entraîner, cette procédure n'intervient jamais dans des phases de calculs intermédiaires.

RÉFÉRENCES

ALEFELD G., HERZBERGER J., (1983), *Introduction to Interval Computations*, New York, Academic Press.

ATTAL K., (1996), "La désaisonnalisation : des origines jusqu'aux nouveaux logiciels X12-ARIMA et TRAMO-SEATS", INSEE-MÉTHODES, *Actes des journées de méthodologie statistique, 11-12 déc. 1996*, N°69-70-71, pp.149-69.

AVIZIENIS A., (1961), "Signed Digit Number Representations for Fast Parallel Arithmetic", *IRE Transactions on Electronic Computers*, N°10, pp.389-400.

BECKETT B., (1990), *Introduction aux méthodes de la cryptologie*, Paris, Masson, Coll.Logique mathématiques informatique, 332 p.

BELL W.R., HILLMER S.C., (1983), "Modelling Times Series with Calendar Variations", *Journal of the American Statistical Association*, Vol.78, N°383, pp.526-35.

BERSTEL J., PIN J.E., POCCHIOLA M., (1991), *Mathématiques et informatique - tome 2 Combinatoire et arithmétique*, Paris, Ediscience international, Coll. Informatique, 257 p. + Programmes.

BOSREDON J., GREGOIRE S., ZAMORA P., (1999), "Données de comptabilité nationale infra-annuelle : quelques problèmes et quelques illustrations", E.ARCHAMBAULT, M.BOEDA (EDS), *Comptabilité nationale - nouvelles frontières*, pp.74-99.

DAUMAS M., MULLER J.M. (EDS), (1997), *Qualité des calculs sur ordinateur - vers des arithmétiques plus fiables ?*, Paris, Masson, Coll.Informatique, 164 p.

FANO R.M., (1961), *Transmission of Information*, New York, J.Wiley and MIT Press.

HERNERT P., (1995), *Les algorithmes*, Paris, PUF, Coll. Que sais-je ?, 128 p.

HUFFMAN D.A., (1952), "A Method for the Construction of Minimum-Redundancy Codes", *Proceedings of IRE*, N°40(9), sept., pp.1098-101.

KNUTH D.E., (1981), *The Art of Programming - tome 2, Seminumerical Algorithms*, Reading (Mass.), Addison-Wesley, 688 p.

LA PORTE M., VIGNES J., (1974), *Algorithmes numériques, analyse et mise en oeuvre - Tome 1, arithmétique des ordinateurs et systèmes linéaires*, Paris, Technip, Coll.Langages et algorithmes de l'informatique, 226 p.

MÉEUS J., (1986), *Calculs astronomiques à l'usage des amateurs*, Paris, Société Astronomique de France, 153 p.

MULLER J.M., (1989), *Arithmétique des ordinateurs - opérateurs et fonctions élémentaires*, Paris, Masson, Coll.Etudes et recherches en informatique, 213 p.

NELSON M., (1993), *La compression des données - textes, images et sons*, Paris, Dunod, Coll.Science informatique, 420 p.+ Programmes.

PICHAT M., VIGNES J., (1993), *Ingéniérie du contrôle de la précision des calculs sur ordinateurs*, Paris, Technip, Coll.Informatique, 233 p. + Programmes.

RIVEST R., SHAMIR A., ADLEMAN L., (1977), "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communication of ACM*, N°21, pp.120-26.

SHANNON C.E., (1951), "Prediction and Entropy of Printed English", *The Bell System Technical Journal*, N°30(1).

VÉLU J., (1994), *Méthodes mathématiques pour l'informatique*, Paris, Dunod, Coll.Science informatique, 452 p.

WIRTH N., (1987), *Algorithmes et structures de données*, Paris, Eyrolles, 320 p.

ZIV J., LAMPEL A., (1977), "A Universal Algorithm for Sequential Data Compression", *IEEE Transactions on Information Theory*, N°23(3), mai, pp.337-43.

ZIV J., LAMPEL A., (1978), "A Compression of Individual Sequences via Variable-Rate Coding", *IEEE Transactions on Information Theory*, N°24(5), sept., pp.530-36.

ANNEXE 4.1 - CALCUL D'ERREUR D'ARRONDI AVEC LE LOGICIEL GNOMBR

Calcul de l'erreur d'arrondi de la fonction Factorielle en virgule flottante[31]

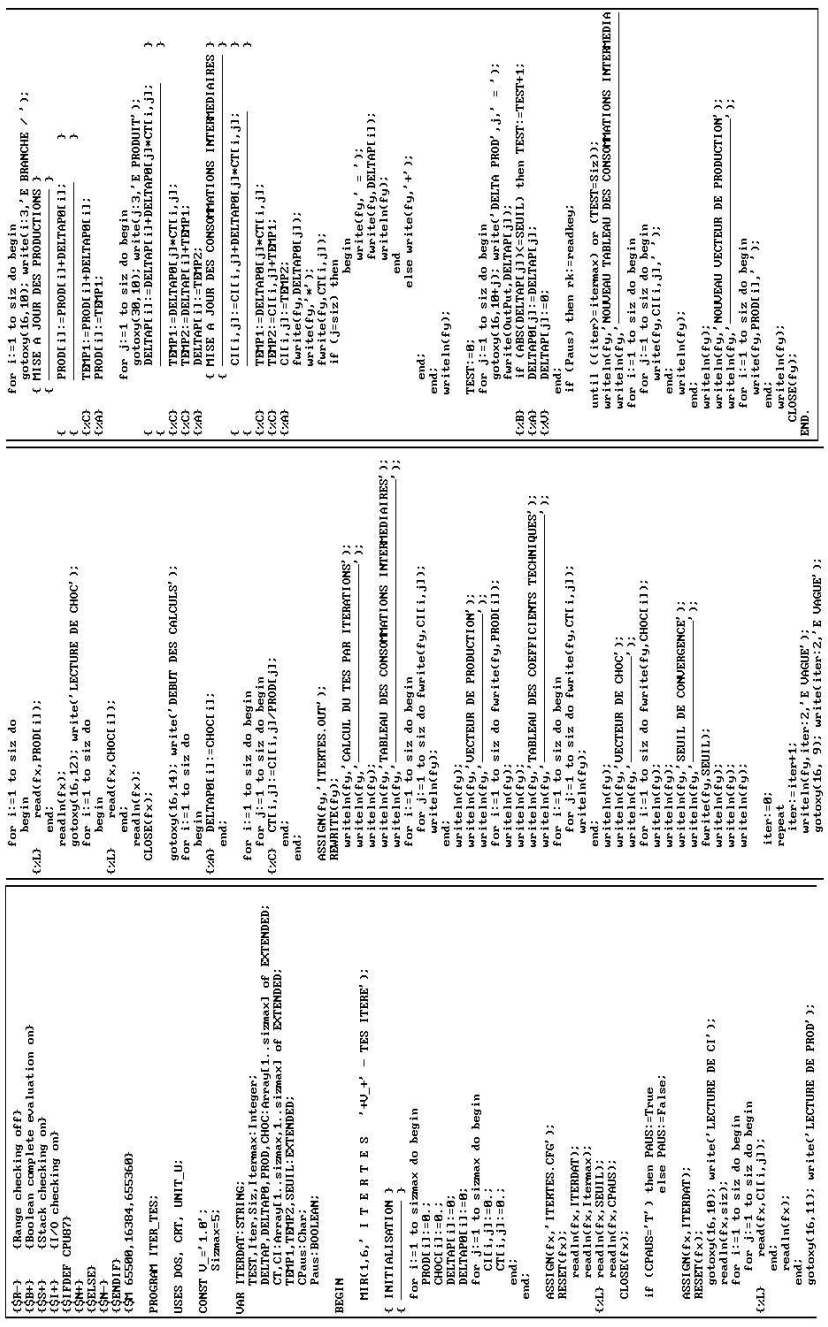
NOMBRES	MULTIPRÉCISION	FLOTTANTE	DIFFERENCE
20 !	2432.0E+00015	2432.0E+00015	0.0E+00000
	90200.0E+00010	90200.0E+00010	
	81766.0E+00005	81766.0E+00005	
	40000.0E+00000	40000.0E+00000	
21 !	51090.0E+00015	51090.0E+00015	40000.0E+00000
	94217.0E+00010	94217.0E+00010	
	17094.0E+00005	17094.0E+00005	
	40000.0E+00000		
22 !	11.0E+00020	11.0E+00020	-20000.0E+00000
	24000.0E+00015	24000.0E+00015	
	72777.0E+00010	72777.0E+00010	
	76076.0E+00005	76077.0E+00005	
	80000.0E+00000		
23 !	258.0E+00020	258.0E+00020	40000.0E+00000
	52016.0E+00015	52016.0E+00015	
	73888.0E+00010	73888.0E+00010	
	49766.0E+00005	49766.0E+00005	
	40000.0E+00000		
24 !	6204.0E+00020	6204.0E+00020	3.0E+00005
	40401.0E+00015	40401.0E+00015	60000.0E+00000
	73323.0E+00010	73323.0E+00010	
	94393.0E+00005	94390.0E+00005	
	60000.0E+00000		
25 !	1.0E+00025	1.0E+00025	-40160.0E+00005
	55112.0E+00020	55112.0E+00020	
	10043.0E+00015	10043.0E+00015	
	33099.0E+00010	33099.0E+00010	
	59840.0E+00005		

NOMBRES	MULTIPRÉCISION	FLOTTANTE	DIFFERENCE
26 !	40.0E+00025	40.0E+00025	-44160.0E+00005
	32914.0E+00020	32914.0E+00020	
	61126.0E+00015	61126.0E+00015	
	60563.0E+00010	60564.0E+00010	
	55840.0E+00005		
27 !	1088.0E+00025	1088.0E+00025	-3.0E+00010
	88694.0E+00020	88694.0E+00020	-92320.0E+00005
	58418.0E+00015	58418.0E+00015	
	35216.0E+00010	35220.0E+00010	
	7680.0E+00005		
28 !	30488.0E+00025	30488.0E+00025	-13949.0E+00010
	83446.0E+00020	83446.0E+00020	-84960.0E+00005
	11713.0E+00015	11714.0E+00015	
	86050.0E+00010		
	15840.0E+00005		
29 !	8.0E+00030	8.0E+00030	-4545.0E+00010
	84176.0E+00025	84176.0E+00025	-63840.0E+00005
	19937.0E+00020	19937.0E+00020	
	39701.0E+00015	39702.0E+00015	
	95454.0E+00010		
	36160.0E+00005		
30 !	265.0E+00030	265.0E+00030	-36369.0E+00010
	25285.0E+00025	25285.0E+00025	-15200.0E+00005
	98121.0E+00020	98121.0E+00020	
	91058.0E+00015	91059.0E+00015	
	63630.0E+00010		
	84800.0E+00005		

[31]- Les résultats en virgule flottante et en multiprécision sont identiques jusqu'à 20! Notre multiplication en multiprécision a été vérifiée au préalable par une procédure de type "preuve par neuf".

ANNEXE 4.2 - APPLICATION DU LOGICIEL GNOMBR À UN TES SIMPLIFIÉ

Programme de calcul de ITERTES avant conversion par EDIT_ GN[32]



[32]- Les symboles {%L}, {%C}, {/A}, {%B} et {%V} (resp.) indiquent qu'il faut traduire des opérations de "lecture", "calcul", "affectation de valeur", de "comparaison" et d'une "affectation entre variables" (resp.).

ANNEXE 4.3 - PROCÉDURE DE REMANTISSAGE INTELLIGIBLE DES RÉSULTATS DE CALCULS

Procédure MANTISSE

```
PROCEDURE MANTISSE(VAR FIL :TEXT;
      REEL :NOMBRE);
VAR II,FORINT,FORFRAC,FORMAN :INTEGER;
      MANTFIX :BOOLEAN;
BEGIN
  MANTFIX :=TRUE;
  FORMAN :=10;
  IF ((ABS(REEL)>=0.000001) AND (ABS(REEL)<100000)) THEN BEGIN
    IF (ABS(REEL)< 0.000001) THEN FORINT :=2;
    IF ((ABS(REEL)>=0.000001) AND (ABS(REEL)<1)) THEN FORINT :=2;
    IF ((ABS(REEL)>=1) AND (ABS(REEL)<10)) THEN FORINT :=2;
    IF ((ABS(REEL)>=10) AND (ABS(REEL)<100)) THEN FORINT :=3;
    IF ((ABS(REEL)>=100) AND (ABS(REEL)<1000)) THEN FORINT :=4;
    IF ((ABS(REEL)>=1000) AND (ABS(REEL)<10000)) THEN FORINT :=5;
    IF ((ABS(REEL)>=10000) AND (ABS(REEL)<100000)) THEN FORINT :=6;
    FORFRAC :=FORMAN-FORINT;
    IF (ABS(REEL)-TRUNC(ABS(REEL))<1.0E-15) THEN BEGIN
      IF (MANTFIX=FALSE) THEN FORFRAC :=0
        ELSE FORFRAC :=10-FORINT;
    END;
    IF (REEL<>0) THEN WRITE(FIL,' ');
    WRITE(FIL,REEL :FORINT :FORFRAC,' ');
    END
  ELSE
    BEGIN
    IF (REEL<>0) THEN WRITE(FIL,REEL :(FORMAN+1),' ')
    ELSE
      BEGIN
        WRITE(FIL,' 0.');
        FOR II :=1 TO FORMAN-2 DO WRITE(FIL,'0');
        WRITE(FIL,' ');
      END;
    END;
END;
```

ANNEXE 4.4 - PRÉSENTATION DES ALGORITHMES SCOLAIRES IMPLÉMENTÉS DANS GBNOMBR

Soient X et Y définis en $Base_b$ comme suit

$$X = \sum_{i=q}^{m} x_i.b^i \qquad avec \;\; 0 \leq x_i \leq b-1$$

$$Y = \sum_{j=p}^{n} y_j.b^i \qquad avec \;\; 0 \leq y_j \leq b-1$$

$$0 \leq q < m \qquad et \;\; 0 \leq p < n$$

1 - Algorithme de l'addition

Soient deux réels X *et* Y, alors l'addition[33] $X+Y$ s'obtient comme suit :

$$X \oplus Y = \sum_{i=Inf(p,q)}^{Sup(n,m)+1} (x_i + y_i + \varepsilon_i - \delta_i).b^i \qquad (\mathbf{1})$$

D'une part, avec $\varepsilon_i = 1$ si $(x_{i-1} + y_{i-1} + \varepsilon_{i-1} - \delta_{i-1}) > b-1$ et avec $\varepsilon_i = 0$ sinon. D'autre part, avec $\delta_i = b$ si $(x_i + y_i + \varepsilon_i) > b-1$ et $\delta_i = 0$ sinon.

2 - Algorithme de la soustraction

Soient deux réels X *et* Y, alors l'addition $X+Y$ s'obtient comme suit :

$$X \ominus Y = \sum_{i=Inf(p,q)}^{Sup(n,m)} (x_i - y_i - \varepsilon_i + \delta_i).b^i \qquad (\mathbf{2})$$

D'une part, avec $\varepsilon_i = 1$ si $(x_{i-1} - y_{i-1}) < 0$ et $\varepsilon_i = 0$ sinon. D'autre part, $\delta_i = b$ si $x_i - y_i - \varepsilon_i < 0$ et $\delta_i = 0$ sinon.

3 - Algorithme de la multiplication

Soient deux réels X *et* Y, alors la multiplication $X*Y$ s'obtient comme suit :

$$X \otimes Y = \sum_{r=0}^{n+m} (S_r + \varepsilon_r - \delta_r).b^r \qquad (\mathbf{3.1})$$

où

$$S_r = \sum_{Inf(r,m)}^{k=Sup(0,r-m)} x_k * y_{r-k} \qquad (\mathbf{3.2})$$

[33]- Les ε_i et δ_i sont des retenues.

D'une part, avec $\varepsilon_i = 1$ si $S_{r-1} + \varepsilon_{r-1} + \delta_{r-1} > b - 1$ et $\varepsilon_r = 0$ sinon et, d'autre part, $\delta_r = b$ si $S_r + \varepsilon_r > b - 1$ et $\delta_r = 0$ sinon.

4 - Algorithme de la division

Soient deux réels X et Y, la division de X par Y équivaut à la relation suivante

$$X = Y.Q + R \tag{4.1}$$

où Q est le quotient et R le reste de la division. Le résultat de la division s'obtient par un processus itératif dont le nombre d'itérations dépend de la précision souhaitée. L'algorithme de Y/X s'obtient comme suit :

$$R_k = \sum_{i=0}^{p^k} (r_i^{k-1} - y_i).b^i \tag{4.2}$$

D'une part avec $y_i = 0 \ \forall \ i \leq p^{k-1} - m$, d'autre part, la valeur initiale est $R_0 = X$ et on a

$$Q_k = b^{p^{k-1}-m} \tag{4.3}$$

A la k-*ème* itération, on a alors deux cas possibles, selon la valeur de R_k :

a) soit $R_k > Y$ alors on réitère avec

$$P^k = Sup_{i \in [0,n]}(i) \tag{4.4}$$

b) soit $R_k \leq Y$ alors on a

$$Q = \sum_{s=0}^{k} Q_s \tag{4.5}$$

V/ Conclusion Générale

Cette présentation nous a amenés à faire plusieurs constatations. Nous avons observé en premier lieu la grande richesse et la grande diversité des algorithmes[1] - toutes les branches des mathématiques sont sollicitées. De plus, il n'y a pas de cloisonnement - Fig.39. Ainsi, par exemple, la résolution de systèmes d'équations relève de la simulation, mais peut faire aussi faire appel à l'optimisation. En second lieu, nous avons constaté la faiblesse des résultats obtenus du fait de l'imprécision structurelle des systèmes de calculs[2], tant logicielle (problème d'approximation, de convergence etc.) que matérielle (arithmétique des machines).

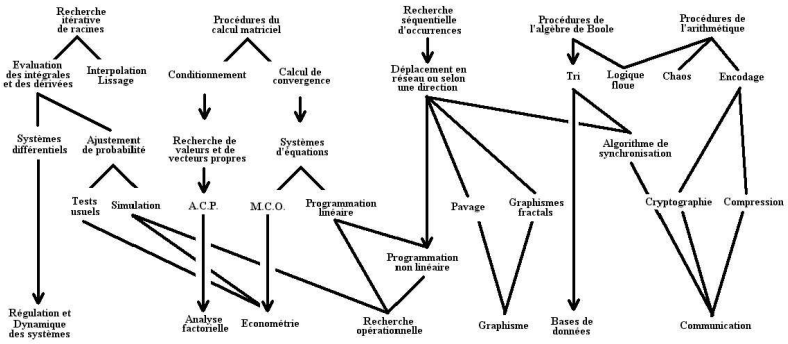


Fig.39 - Des algorithmes basiques aux macro-algorithmes

Pourtant, nous avons noté des paradoxes bien plus déroutants encore. Alors que nous avions dit que le modélisateur ne pouvait à proprement parler, simuler de marche aléatoire, il est symptomatique de relever à propos des arithmétiques stochastiques que "*Le principe fondamental consiste donc, à interpréter les* α_i *(quantités perdues) non plus comme des quantités déterministes mais comme des quantités aléatoires.* [...] *Le résultat informatique apparaît alors comme une quantité aléatoire*" (M.PICHAT et J.VIGNES, *op.cit.*, pp.98-99). On mesure ici l'importance de cette phrase, lorsque le travail des économètres consiste précisément à "*extraire*" la partie non aléatoire de résultats d'estimation[3]. De même, l'alternative entre modélisation "centralisée" et modélisation "décentralisée" ne serait-elle pas une fausse alternative ? Certes la critique autrichienne, bien résumée par L.MISES (1935), selon laquelle le réalité économique et sociale ne peut être représentée par des équations - du fait de l'essence subjective des relations inter-individuelles - a remis en cause l'efficacité des procédures cen-

[1]- Concernant les langages à l'usage des économistes modélisateurs, D.A.KENDRICK & H.M.AMMAN (1999) les classent en trois catégories. Les high level languages tels que GAUSS, GAMS, Mathematica, Maple ou MATLAB, les low level languages tels que FORTRAN, Basic, C/C++ ou Java enfin, les langages spécialisés dans le développement d'interfaces graphiques et de communication tels que Visual Basic, C/C++ ou Java.

[2]- A.TURING ("On Computable Numbers", *Proceedings of the Mathematical Society*, N°42, 1936, pp.230-65) a démontré que certains nombres étaient définissables mais non calculables par des machines.

[3]- Ce fait pourrait peut être en partie expliquer le paradoxe bien connu des modélisateurs, à savoir que l'affinement des modèles (par augmentation du nombre de variables, de dimensions etc.) n'accroît pas nécessairement la précision des résultats (P.ARTUS et al., *op.cit.*, 1986, pp.243-44).

tralisées de calcul économique. Mais l'expression modélisation décentralisée[4] a-t-elle un sens ? En effet par définition, une modélisation ne peut prendre en compte qu'un seul point de vue - celui de l'algorithme qui régit le système. Elle ne peut prétendre simuler le point de vue d'une foule d'individus qui, dans la réalité, effectuent chacun de leur côté leur propre calcul économique. Il nous semble qu'il y a là substitution d'une procédure "nécessaire" à un ensemble d'actions "contingentes" - voir notre communication (2000.b). On serait dès lors raisonnablement fondés à se poser la question Jusqu'où, dans ces conditions, une procédure numérique peut-elle valider une théorie économique ? En fait, l'algorithmique intervient dans la démarche de modélisation économique, à un niveau bien plus théorique qu'on ne pourrait le penser : la démarche de représentation de l'économie par des algorithmes constitue une hypothèse économique[5]. Pour autant, il ne faudrait pas disqualifier la démarche algorithmique dans la construction et/ou la validation des théories économiques. Face à ces mêmes indéterminations, les physiciens disposent des outils de contrôle éprouvés que sont l'expérimentation et la métrologie. Qu'en est-il de l'économiste ? Certes, d'une part, la Comptabilité nationale relève peu ou prou de la démarche métrologique[6], mais sans doute n'atteindra-t-elle jamais le degré de fiabilité (si relatif soit-il) de la métrologie physique : tout n'est pas observable[7] - et ce qui l'est, n'est pas nécessairement fiable (O.MORGENSTERN, 1950). D'autre part, l'économie expérimentale, au départ comme champ d'application de la théorie des jeux, s'est progressivement révélée être une branche fort utile pour valider ou invalider certains schémas théoriques[8]. Il faudrait, nous semble-t-il, tester des formalisation plus analogiques des comportements[9]. En tout état de cause, le recours à l'algorithmique requiert un contrôle minutieux dans l'élaboration des systèmes et une forte cohérence des processus algorithmiques avec les schémas théoriques eux-mêmes.

---

[4]- Certains chercheurs de la Computational Economics (voir §III.b.ii ; citons également N.J.VRIEND, 1999) voient dans ce type de modélisation une amélioration de la réprésentation de la complexité sociale.

[5]- Voir B.PAULRÉ (1985) à propos des risques de confusion liés à l'utilisation de l'inférence statistique comme mode de représentation causale.

[6]- Plus encore qu'en physique la mesure statistique dépend de la qualité des instruments voir O.ARKHIPOFF ("Fiabilité des comptes : une étude par simulation", E.ARCHAMBAULT, O.ARKHIPOFF (EDS), *La comptabilité nationale pourquoi faire ?*, Paris, Economica, 1992, pp.141-49 ; "Chap.11 - L'information économique et sociale", G.PRIEUR, M.NADI (EDS), *La mesure et l'instrumentation - état de l'art et perspectives*, Paris, Masson, Collection Mesures physiques, 1995, pp.303-24.).

[7]- Le statisticien n'a pas nécessairement accès aux vraies valeurs (R.STONE, "Adjusting the National Accounts", *Communication Istituto Centrale di Statistica*, Rome, 26 sept., 1988, 33 p.).

[8]- L'hypothèse d'ajustement walrasien a été invalidée par l'expérience alors que l'hypothèse hayekienne de coordination avec des petits effectifs a été confirmée (V.L.SMITH, "An Experimental Study of Competitive Market Behavior", *Journal of Political Economy*, N°70, 1962, pp.111-37).

[9]- Dans nos communications (1999.b) et (2000.b) (resp.) d'économie expérimentale et d'économie autrichienne (resp.), nous basant sur une critique de la modélisation économétrique et de la planification, nous avons élaboré un système de modélisation individualiste méthodologique qui peut être validé par expérimentation (le couple de logiciels `SINGUL-ECHANGE` ©). Mais nous avons également conclu qu'il n'y avait pas de panacée en la matière : le modèle SINGUL-Généralisé est une chimère.

RÉFÉRENCES

KENDRICK D.A, AMMAN H.M., (1999), "Programming Languages in Economics", *Computational Economics*, N°14, pp.151-181.

MORGENSTERN O., (1950), *On the Accuracy of Economic Observations*, Princeton, Princeton UP, (trad.1972, Paris, Dunod).

MISES L.(Von), (1938), "Les équations de l'économie mathématique et le problème du calcul économique en régime socialiste", *Revue d'économie politique*, N°97(6), pp.899-906, (Réimpr.1987).

PAULRÉ B., (1985), *La causalité en économie - signification et portée de la modélisation structurelle*, Lyon, Presses universitaires de Lyon, Coll.Science des systèmes, 440 p.

PICHAT M., VIGNES J., (1993), *Ingéniérie du contrôle de la précision des calculs sur ordinateurs*, Paris, Technip, Coll.Informatique, 233 p. + Programmes.

VRIEND N.J., (1999), "Was Hayek an ACE[10]", *Working Paper, Queen Mary and Westfield College*, University of London, 34 p.

[10]- L'auteur fait ici un jeu de mot : ACE signifie Agent-based Computational Economics.

ANNEXE 5 - LES ÉCONOMISTES ET LES MACHINES À CALCULER

Le MONIAC de A.W.PHILLIPS[11]

[11]- Pour enseigner la théorie keynésienne, A.W.PHILLIPS conçut une machine analogique, le Moniac ("Mechanical Models in Economic Dynamics", *Economica*, N°17, 1950, pp.283-305).

LOGICAL MACHINE de W.S.JEVONS[12]

[12]- Le co-fondateur de l'Ecole marginaliste, W.S.JEVONS exerça aussi une carrière de logicien (*Elementary Lessons in Logic*, London, 1876).