Contribution à l'étude de méthodes de contrôle automatique de l'erreur d'arrondi : la méthodologie SCALP

Philippe François

THESE

Présentée par
Philippe FRANCOIS

Pour obtenir le titre de ***DOCTEUR*** de
L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(Arrêté ministériel du 23 novembre 1988)

Spécialité : **Mathématiques Appliquées**

Contribution à l'étude de méthodes de contrôle automatique de l'erreur d'arrondi : la méthodologie SCALP.

Thèse soutenue le 19 Décembre 1989.

Composition du jury :

J.DELLA DORA	Président
M.COSNARD	rapporteur
B.PHILIPPE	rapporteur
J-M MULLER	
J.VIGNES	
P. DORIO	
P-J. LAURENT	

Thèse préparée au sein du laboratoire TIM3, Unité.Associée. au CNRS n°397.

Joignez ce qui est complet et ce qui ne l'est pas, ce qui concorde et ce qui discorde, ce qui est en harmonie et ce qui est est en désaccord.

Héraclite.

Peut-être y a t-il d'autres connaissances à acquérir, d'autres interrogations à poser aujourd'hui, en particulier, non de ce que d'autres ont su, mais de ce qu'ils ont ignoré.

S.Moscovici.

It is probably true quite generally that in the history of human thinking the most fruitful developments frequently take place at those points where two differents line of thought meeet. These lines may have there roots in quite different cultural environments or different religious traditions : hence if they actually meet, that is if they are at least so much related to each other that a real interaction can take place, then one may hope that new and interesting developements may follow.

W.Heisenberg.

A Giovanna ,

A mes parents qui m'ont permis d'en arriver là.

Je voudrais exprimer ma profonde reconnaissance à J-M.MULLER, Chargé de recherche au CNRS, qui a dirigé cette thèse. Outre sa science du calcul sur ordinateur; ses conseils, son humour caustique, son art des contre exemples et surtout la confiance qu'il a su me témoigner, m'ont permis de mener à bien ce travail.

J'adresse de vifs remerciements à M.COSNARD, Professeur à L'ENS de Lyon, qui a accepté d'être le rapporteur de cette thèse et a, malgré ses nombreuses occupations, toujours suivi mon travail avec attention. Outre l'étendue de ses connaissances scientifiques, j'ai beaucoup apprécié sa chaleur humaine.

Je remercie également chaleureusement J.VIGNES, Professeur à l'université de PARIS 6 pour l'honneur qu'il me fait en participant à ce jury et pour la gentillesse avec laquelle il a bien voulu répondre à nos questions sur CESTAC.

Je tiens aussi à remercier J.DELLA DORA, Professeur à L'INP Grenoble qui a dirigé mon DEA, de me faire l'honneur de présider le Jury de cette thèse et d'y représenter le calcul formel, autre alternative non négligeable aux méthodes de contrôle des erreurs numériques abordées dans ce travail.

Toute ma gratitude va également à P-J.LAURENT, Professeur à l'université Joseph Fourier de Grenoble pour l'honneur qu'il me fait en examinant cette thèse et pour l'occasion qu'il m'a donnée d'enseigner l'optimisation en deuxième année de L'ENSIMAG.

Je remercie aussi B.PHILIPPE, Chargé de recherche à L'INRIA de Rennes, d'avoir accepté d'être le rapporteur de cette thèse et de s'être ainsi intéressé à mon travail.

Enfin, je voudrais particulièrement remercier Madame DORIO, Chef du service de mathématiques spatiales et d'analyse numérique du Centre National d'Etudes Spatiales de Toulouse, qui est à l'origine de cette étude et a su me faire partager le point de vue de l'industriel sur la qualité numérique .

Je ne voudrais pas oublier :

JC.Bergès, Ingénieur au CNES, pour l'attention avec laquelle il a suivi mon travail;

Y.Herreros, Roi de l'assembleur 68000 (et autres) sans qui la partie perturbation automatique du logiciel implantant la méthodologie présentée dans cette thèse n'aurait pu naitre;

A.Douet et F.Gerard dont j'ai eu le plaisir de coencadrer le Projet d'ingénieur ENSIMAG avec JM.Muller et qui m'ont permis de disposer d'un logiciel convivial de représentation d'histogramme pour mon travail;

et L.Crouzet qui a effectué dans le cadre de son DEA de nombreuses expérimentations sur les méthodes de perturbation.

Merci également à tous les membres de l'équipe d'Algorithmique Parallèle et de calcul formel pour leur ambiance de travail sympathique et leurs orgies de gateaux.

SOMMAIRE

Chapitre V Stabilité et Conditionnement Arithmétique des Logiciels par Perturbation : SCALP.

INTRODUCTION

"It makes me nervous to fly on airplanes since I know they are designed using floating point arithmetic "
A.Housholder.

• ***Des erreurs d'arrondi.***

Comme on le sait, les calculs sur ordinateurs s'effectuent par arrondis successifs. De ce fait, les résultats numériques fournis par une machine sont entachés d'une "***erreur arithmétique***". Les deux exemples suivants illustrent l'importance que peut avoir ce phénomène.

Exemple 1 [FMM] :

On cherche à calculer $I_n = \int_0^1 x^n e^{-x} dx$ pour n = 50. Pour ceci, on peut remarquer en intégrant par parties que I_n est égal à $nI_{n-1} - 1/e$. La première idée qui vient à l'esprit est alors de partir de $I_0 = 1 - 1/e$, et de calculer successivement : $I_1 = I_0 - 1/e$, $I_2 = 2I_1 - 1/e$,..., $I_{50} = 50\, I_{49} - 1/e$.

La table 1 décrit les résultats obtenus... ils sont catastrophiques. L'explication de ce phénomène est très simple : il suffit de regarder la formule de récurrence $I_n = nI_{n-1} - 1/e$ pour constater que lorsqu'on passe de I_{n-1} à I_n, l'erreur sur I_{n-1} est approximativement multipliée par n. On en déduit que l'erreur sur I_n croit comme la factorielle de n.

Il est facile de remédier à ceci : supposons maintenant que l'on connaisse I_p, p>50, et que l'on calcule successivement I_{p-1}, I_{p-2},..., I_{50} à l'aide de la relation réciproque $I_{n-1} = (I_n + 1/e)/n$. En passant de I_n à I_{n-1}, l'erreur sera approximativement divisée par n. L'erreur sur I_{50} sera donc proportionnelle à 50!/p!. On voit qu'alors il n'est même pas nécessaire de connaître I_p : en effet, si p est suffisamment grand, on peut prendre à la place de I_p n'importe quelle valeur comprise entre 0 et 1, de toute façon l'erreur commise sur I_p sera inférieure à 1. Si par exemple, on choisit p = 60, et si on remplace I_p par 1, l'erreur commise sur I_{50} sera de l'ordre (en négligeant tout de même les arrondis) de $50!/60! = 3.10^{-18}$. La table 2 donne les valeurs obtenues en effectuant cette "itération rétrograde". Il est piquant de constater qu'en partant d'une valeur correcte ($I_0 = 1 - 1/e$), on aboutit à cause des erreurs d'arrondi à un résultat faux de plusieurs ordres de grandeur, tandis qu'en partant d'une valeur parfaitement fantaisiste (on remplace I_{60} par 1), on obtient un résultat exact à la précision machine près.

	Valeur calculée	Valeur exacte
0 :	6.32120558828558e-01	
1 :	2.64241117657115e-01	
2 :	1.60602794142788e-01	
3 :	1.13928941256923e-01	1.13928941256923e-01
4 :	8.78363238562483e-02	
5 :	7.13021781097991e-02	7.13021781098032e-02
6 :	5.99336274873523e-02	
7 :	5.16559512400239e-02	
8 :	4.53681687487486e-02	
9 :	4.04340775672951e-02	
10 :	3.64613345015088e-02	3.64613346241073e-02
11 :	3.31952383451544e-02	
12 :	3.04634189704100e-02	
13 :	2.81450054438883e-02	
14 :	2.61506350429941e-02	
15 :	2.43800844734690e-02	
16 :	2.22019104040609e-02	
17 :	9.55303569759369e-03	
18 :	-1.95924798614756e-01	
19 :	-4.09045061485180e+00	
20 :	-8.21768917382075e+01	1.83504676972562e-02
21 :	-1.72608260594353e+03	
	...	
29 :	-2.98717649194210e+14	
30 :	-8.96152947582629e+15	1.22495029578589e-02
31 :	-2.77807413750615e+17	
	...	
48 :	-4.19402341492941e+44	
49 :	-2.05507147331541e+46	
50 :	-1.02753573665770e+48	7.35470679580019e-03

Table 1. Calcul de I_n par itération directe.

60 :	1.0000000000000000e+0
59 :	2.279799068619071e-2
58 :	6.621651387417509e-3
57 :	6.456915388945859e-3
56 :	6.567304501059442e-3
55 :	6.686549029866103e-3
54 :	6.810290730932880e-3
53 :	6.938698738932874e-3
52 :	7.072040375667457e-3
51 :	7.210605414367496e-3
50 :	7.354706795800192e-3

Table 2. calcul de I_n par itération rétrograde.

L'exemple précédent montre que les résultats d'un calcul numérique peuvent être entachés d'une erreur énorme sans que l'on soit nécessairement au voisinage d'un dépassement de capacité, et au bout d'un très petit nombre d'opérations arithmétiques. L'erreur obtenue sur I_{50} n'est toutefois, en un certain sens, pas très grave, car elle est flagrante : il suffit de regarder les nombres de la table 1 pour constater "qu'il se passe quelque chose". Ce n'est hélas pas toujours le cas : l'exemple suivant montre que même lorsqu'en machine on observe une "bonne convergence", qui semble rapide, sans qu'aucun résultat intermédiaire ne s'approche de zéro ou de l'infini (risques de dépassement de capacité), le résultat observé est sujet à caution.

Exemple 2 [Mul] :
Considérons la suite (a_n) définie comme suit :

$$a_{n+1} = 111 - \frac{1130}{a_n} + \frac{3000}{a_n a_{n-1}}$$

$$a_0 = 11/2, \qquad a_1 = 61/11$$

La limite de a_n est égale à 6. Pourtant, sur *n'importe quelle machine*, on observera une convergence *rapide* de cette suite vers 100... la table 3 présente les valeurs des premiers termes de cette suite, calculées sur SUN3 en double précision. Les valeurs exactes ont été calculées à l'aide du système de calcul formel REDUCE.

	Valeur calculée	Valeur exacte
0 :	5.50000000000000e+00	
1 :	5.54545454545455e+00	
2 :	5.59016393442624e+00	
3 :	5.63343108504413e+00	
4 :	5.67464862051261e+00	
5 :	5.71332905242392e+00	
6 :	5.74912092046204e+00	
7 :	5.78181093369010e+00	
8 :	5.81131446660218e+00	
9 :	5.83766047654396e+00	
10 :	5.86101878599628e+00	
11 :	5.88252460826931e+00	
12 :	5.91865532380549e+00	$\frac{84467679721}{14281397141} = 5.9145249...$
13 :	6.24396181530611e+00	
14 :	1.12030873728409e+00	$\frac{2973697798081}{500702562701} = 5.9390505...$
15 :	5.30217126449968e+01	
16 :	9.47384227927645e+01	
17 :	9.96696508735507e+01	
18 :	9.99802577609368e+01	
19 :	9.99988224533759e+01	
20 :	9.99999297074558e+01	
21 :	9.99999958004987e+01	
22 :	9.99999997489326e+01	
23 :	9.99999999849811e+01	
24 :	9.99999999991011e+01	
25 :	9.99999999999462e+01	
26 :	9.99999999999968e+01	
27 :	9.99999999999998e+01	
28 :	1.00000000000000e+02	
29 :	1.00000000000000e+02	
30 :	1.00000000000000e+02	5.999999989...

Table 3. Valeurs calculées et valeurs exactes des termes de la suite a_n.

• ***De l'importance pratique grandissante de ces erreurs :***

L'influence des arrondis sur la signification du résultat d'un calcul a été envisagée dès la fin du siècle dernier par des astronomes [Bro], [Sch], [Eke] et dès les débuts des machines électroniques par des pionniers de l'informatique comme Turing [Tur], Von Neuman [VG]. Cependant, ce problème n'est encore qu'assez peu pris en compte par la plupart des utilisateurs qui considèrent généralement que l'erreur arithmétique est tout à fait négligeable devant les autres approximations faites au cours de leurs calculs (erreurs de modélisation, erreurs de méthodes, erreurs de mesures).

Pourtant, pour plusieurs raisons technico-économiques cette question semble devoir devenir importante dans les applications [Leg], [Ber], [Dor]. Tout d'abord, sous la pression de la demande, les modèles mathématiques effectivement utilisés arrivent désormais souvent à une précision proche de celle de la machine. Les erreurs dues aux arrondis effectués en machine (erreurs de discrétisation) perturbent donc de manière plus significative la qualité des produits. Par ailleurs, dans certains domaines de pointe comme le "calcul embarqué", les contraintes économiques et techniques limitent la taille des calculateurs (donc, la précision) et accentuent les problèmes d'erreur arithmétique. Enfin, l'utilisation accrue des supers calculateurs pour gagner du temps et/ou traiter des problèmes de plus grande taille accroit elle aussi les problèmes dus à la discrétisation-machine. D'une part parce que la probabilité d'avoir des accidents arithmétiques augmente avec la longueur des calculs. D'autre part parce que certaines techniques d'adaptation d'algorithmes aux nouveaux ordinateurs (notamment de parallélisation) dans un objectif de gain de temps semblent amplifier les problèmes d'instabilité [Sam]. Parallèlement, l'utilisation depuis peu de différents ordinateurs pour le développement fait prendre conscience aux concepteurs de logiciels de la contingence des résultats d'un calcul sur ordinateur. La recherche d'outils de contrôle de l'erreur arithmétique est donc d'actualité. Le contexte du travail décrit ici en est d'ailleurs une bonne illustration puisqu'il a été réalisé dans le cadre d'un contrat de recherche sur ce problème entre le Centre National d'Etudes Spatiales et L'Institut National Polytechnique de Grenoble.

• ***Quels principes de validation du logiciel numérique ?***

Les spécifications d'une méthode de qualification du logiciel peuvent être très vastes. Elles dépendent évidemment des degrés d'exigence et de réalisme de l'utilisateur. Tout le monde voudrait disposer d'une méthode de correction de l'erreur arithmétique. Le moindre qu'on semble devoir exiger est de pouvoir décider si un logiciel est fiable. La plupart des méthodes existantes (méthode de Miller [Mil], arithmétiques d'intervalles [Lan], [KM], CESTAC[BC], [PV]) se proposent d'estimer l'erreur d'arrondi. Mais on peut aussi chercher à obtenir des renseignements plus qualitatifs sur cette erreur. On peut par exemple vouloir mesurer la qualité d'un logiciel au regard de la difficulté du problème qu'il résout, chercher à mesurer l'effet d'un changement de précision des calculs intermédiaires ou des données sur le résultat, chercher à comparer deux algorithmes de résolution d'un même problème. C'est une voie de recherche assez peu explorée et qui semble pourtant intéressante. Elle a constitué le pôle d'intérêt principal de notre travail.

• ***Approche adoptée.***

Nous nous sommes fixés l'objectif de déboucher sur une méthodologie d'analyse expérimentale des programmes (du point de vue de l'erreur arithmétique) basée sur un cadre théorique clair de définition de la qualité arithmétique. Dans cette optique, l'idée ancienne [For] de sonder statistiquement la résistance aux arrondis (comme on regarde si un système mécanique est stable en le secouant) en perturbant les calculs, qui est déjà à la base de différentes méthodes de contrôle de l'erreur d'arrondi, en particulier la méthode CESTAC de J Vignes [BC], [PV], nous a paru intéressante à développer. D'abord parce que, d'un point de vue général, perturber un problème semble un moyen de déployer ses singularités [Tho], ensuite parce que certaines perturbations des calculs peuvent s'implanter facilement à un coût "modeste". Par ailleurs, nous étions dans un contexte favorable pour effectuer une telle étude puisque la recherche de la communauté Française sur l'utilisation des perturbations était dans une phase très active. D'un point de vue expérimental, nous avons cherché à tirer parti de l'utilisation de différents types de perturbations. Pour ce qui est du cadre théorique, il paraissait important de faire une bonne synthèse de la littérature et de se référer à d'autres cadres d'analyse de la stabilité. Nous nous sommes en particulier inspirés de certains principes d'analyse de schémas de discrétisation ou d'itérations. Nous devons également un certain nombre de pistes aux travaux de l'équipe de F.Chatelin [Cha] et au modèle de Chesneaux d'un calcul en arithmétique perturbée [Che].

• ***Plan du travail.***

Le **chapitre I** est une introduction au problème des erreurs d'arrondi. On visualise d'abord par quelques exemples la variété et la gravité des problèmes posés par ces erreurs. On souligne particulièrement le fait que *les arrondis ne se contentent pas d'altérer la valeur des solutions mais peuvent aussi détruire la structure mathématique des phénomènes simulés.* On examine ensuite succintement quelques réponses théoriques possibles au niveau de la conception des systèmes arithmétiques et de l'écriture des programmes. On montre en particulier comment un choix adéquat de *la représentation des nombres permet de respecter certaines propriétés algébriques ou topologiques des réels.*

Dans le **chapitre II**, on examine une partie du grand nombre des méthodes d'évaluation d'erreur d'arrondi qui bien que peu unifiées et la plupart du temps empiriques peuvent se répartir en deux classes : les méthodes statistiques et les méthodes déterministes. Les approches statistiques et en particulier les méthodes basées sur des perturbations aléatoires des calculs ont retenu plus spécifiquement notre attention. On souligne à la fois l'intérêt expérimental d'une approche probabiliste et les *difficultés de fondement posées par les approches paramétriques* notamment à cause des spécificités de l'arithmétique ou de l'existence de biais ou de non linéarités.

Dans la suite du travail, on s'intéresse, tout comme récemment F.Chatelin [Cha] à l'utilisation non paramétrique de perturbations des calculs pour l'analyse de la résistance aux arrondis (la stabilité arithmétique) des programmes.

On introduit d'abord dans le **chapitre III** un cadre théorique pour l'étude de la stabilité arithmétique. Les notions présentées permettent de donner un sens autre qu'empirique à la notion de qualité d'un logiciel numérique. Le nouveau concept de régularité permet par exemple d'expliciter ce qu'on peut entendre par "l'information gagnée sur le résultat par bit de précision supplémentaire ". Les *mesures de stabilité* définies permettent également de faire le lien entre la qualité arithmétique d'un logiciel et la difficulté du problème qu'il résout. Par ailleurs, le cadre théorique introduit autorise une approche unifiée de la stabilité arithmétique et de la stabilité numérique.

Dans le **chapitre IV**, on donne la spécification de l'outil logiciel qui a servi de support à notre étude et que nous avons fourni au CNES. Son originalité par rapport à d'autres outils est de simuler un grand nombre de perturbations différentes et d'autoriser deux niveaux de perturbations (perturbations des données et/ou perturbation des opérateurs).

Enfin, dans le **chapitre V**, on présente notre méthodologie "SCALP" d'utilisation de ce logiciel pour la qualification de certains programmes de calcul à travers les estimations (sous des hypothèses qui sont précisées) des indices de qualité du chapitre III.

BIBLIOGRAPHIE

[Ber] J C BERGES, Communication aux journées de L'INRIA sur la précision de l'arithmétique en calcul scientifique, Rennes, Avril 1989.

[Bro] D BROUWER, On the accumulation of errors in numerical integration. Astron J, vol 46, p149-153, 1937.

[BC] M C BRUNET, F CHATELIN, CESTAC, a tool for a stochastic round off error analysis. Numerical Mathematics and Applications, IMACS, 1986.

[Cha] F CHATELIN, Analyse statistique de la qualité numérique et arithmétique de la résolution approchée d'équations par calcul sur ordinateur. Etude F. 133, centre scientifique IBM FRANCE, avril 1988.

[Che] JM CHESNEAUX, Etude théorique et implantation en Ada de la méthode CESTAC. Thèse de l'Université Paris 6, avril 1988.

[Dor] P DORIO, Le calcul scientifique. Qualité numérique et logiciel scientifique, Séminaire du projet aquarel organisé par le département de mathématiques appliquées et d'analyse numérique, Recueil des communications, CNES, Paris, Octobre 1989.

[Ekc] I EKELAND, Le calcul, l'imprevu. Collection science ouverte, editions du seuil, Paris, 1984.

[FMM] FORSYTHE, MALCOM, MOLER, Computer methods for math computations. Prentice-Hall, 1974.

[KM] U W.KULISCH and W L.MIRANKER, Computer arithmetic in theory and practice. Academic Press, New York, 1981.

[Lan] E LANGE, Implementation and test of the ACRITH facility in a system 370. IEEE trans of computers, vol C36, N°9, September 1987.

[Leg] M LEGENDRE, Les mathématiques appliquées à la simulation numérique. Le courrier du CNRS, Images des mathématiques, 1985.

[Mil] W MILLER, Software for the roundoff analysis. ACM trans on Math Software, vol 1, pp 524-547, 1975.

[Mul] JM MULLER, Arithmétique des ordinateurs, Masson, 1989.

[PV] M La PORTE et J VIGNES, Error analysis in computing. Information processing, 1974.

[Sam] A SAMEH, Numerical parallel algorithms, a survey. In high speed computer and algorithms organization. D Kuck, D Lawrie and A Sameh eds, pp 207-228, Academic Press, 1977.

[Sch] F SCHLESINGER, On the errors in the sum of tabular quantities. Astronomical Journal, vol 30, p183-190, 1917.

[Tho] R THOM, Modèles mathématiques de la morphogénèse. Editions Christian Bourgeois, Paris, 1981.

[Tur] AM TURING, Rounding off errors in matrix processes. Quat J Rech, vol1, 287-308, 1948.

[VG] J VON NEUMANN, HH GOLDSTEIN, Numerical inverting of high order. I. Proc Amer Soc, vol 53, 1021-1099, 1947.

CHAPITRE I

SIMULATION DES CALCULS PAR DES ARITHMETIQUES DISCRETES

INTRODUCTION

Bien qu'une machine ne puisse effectuer qu'un nombre fini de transformations sur des objets discrets, la plupart des scientifiques cherchent à représenter et à traiter informatiquement des structures continues et en particulier le corps des réels et les opérateurs sur ce corps (opérateurs algébriques, limites). Une telle représentation du continu mathématique par une structure discrète n'est évidemment qu'approchée. Chaque calcul produit par une machine est entaché d'erreurs d'arrondi. Pire, comme nous le verrons, le passage d'un algorithme sur un ordinateur change parfois sa nature mathématique (cf [Rob]).

Il importe donc de penser le choix des représentations machine du continu à la lumière de ce problème. L'enjeu est double. Il s'agit d'abord de trouver des principes capables de rationnaliser le choix d'une arithmétique pour aboutir à des systèmes compatibles avec les désirs des utilisateurs. Il s'agit ensuite de comprendre comment se pose le problème du passage du continu au discret afin de mieux cerner le phénomène de l'erreur arithmétique et d'aider à l'utilisation des calculateurs.

De nombreux auteurs ont réfléchi sur le sujet, mais il n'existe pas véritablement de théorie de l'approximation du continu par le discret. La plupart des auteurs étudient uniquement l'arithmétique flottante ou se limitent aux aspects quantitatifs. Cela tient à notre avis à deux raisons essentielles. D'abord à la difficulté mathématique du sujet : il semble que les outils pour traiter ce problème soient encore à naître. Ensuite aux sujets de prédilection des arithméticiens des ordinateurs traditionnellement plus préoccupés du problème d'optimisation du temps de calcul que de la recherche d'arithmétiques qualitatives. On présente ici quelques réflexions sur ce thème en guise de sensibilisation aux phénomènes d'erreurs d'arrondi et de présentation de l'arithmétique des ordinateurs.

On esquisse d'abord un modèle général simple d'un système arithmétique machine. Après avoir donné la représentation des algorithmes qui sera adoptée dans cette thèse, on développe alors ce qui nous semble devoir être la base d'une spécification sur ce modèle d'une arithmétique efficace du point de vue de la qualité de la représentation des calculs. On présente ensuite les modèles usuels d'arithmétiques machine ainsi que l'arithmétique flottante qui est de loin la plus utilisée en pratique.

Dans une deuxième partie, on montre un certain nombre de problèmes pouvant être posés par l'arithmétique flottante. On espère ainsi sensibiliser le lecteur à l'importance de réfléchir à l'erreur d'arrondi, y compris en amont, au niveau de la conception des systèmes arithmétiques.

Dans une troisième partie, on essaie de présenter les nombreux systèmes arithmétiques qui ont été proposés en vue de limiter l'erreur d'arrondi. La possibilité de rechercher systématiquement de nouveaux systèmes de représentation des réels à partir de spécifications constitue le fil conducteur de cette présentation. Cette approche n'a nous le croyons jamais été adoptée. Pour illustrer ce qu'elle pourrait être, *on donne à titre d'exemple deux théorèmes nouveaux spécifiant complètement les systèmes "usuels" vérifiant des axiomes algébriques simples. On montre également comment on peut dériver l'arithmétique d'intervalle et l'arithmétique stochastique de Vignes d'une même arithmétique ensembliste dotée de propriétés d'inf continuité.*

Enfin, dans une dernière partie, on discute très rapidement la possibilité de minimiser automatiquement l'occurrence des instabilités arithmétiques dans un calcul en arithmétique virgule flottante dès la programmation.

A-REPRESENTATION DES ALGORITHMES DE CALCUL EN MACHINE

Nous précisons d'abord les concepts d'arithmétique machine et de simulation adoptés dans la thèse. La présentation est volontairement très générale pour pouvoir aborder dans un cadre unique et avec un certain recul tous les problèmes étudiés.

A-1 Modèle d'arithmétique d'ordinateur ?

Du point de vue de l'utilisateur, une arithmétique d'ordinateur se caractérise mathématiquement par la donnée :

- D'un ensemble fini S de "nombres machine".
- D'un ensemble fini $\mathcal{F}$ d'opérateurs sur S disponibles sur la machine et qu'on appelera *opérateurs machine élémentaires.*
- D'un procédé de représentation τ de la structure des réels en machine qui vérifie :

  $\forall\ x \in \mathbb{R}\ \tau(x) \in S$

  $\forall$ f opérateur à valeurs réelles utilisé, $\tau(f)$ est un produit de composition d'éléments de $\mathcal{F}$.

(Tout opérateur de calcul doit pouvoir se représenter en machine à l'aide d'opérateurs machine élémentaires)

- D'une interprétation des résultats i : $S \rightarrow \mathcal{P}(\mathbb{R})$ telle que $\tau \circ i$ soit égal à l' identité.

Mais de nombreuses questions se posent :

Quels nombres représenter ? Quels opérateurs élémentaires choisir ? S doit t-il être une partie de IR ? τ doit t-il être déterministe?

A-2 Qu'imposer à une arithmétique machine ?

L'idéal serait de disposer d'un isomorphisme de structures. Malheureusement, il n'en existe pas. On cherchera à s'en approcher au maximum. Mais selon quels critères?

Les utilisateurs de la simulation numérique conçoivent des algorithmes surIR pour résoudre leurs problèmes continus et les exécutent en machine. Une arithmétique machine doit donc être capable de représenter le comportement d'algorithmes mathématiques sur IR. C'est ce que nous entendrons par capacité de simulation du continu réel.

A-2-1 Algorithmes mathématiques

Un **algorithme mathématique** sur un ensemble E est empiriquement une suite de transformations dans E, il semble donc tout naturel de représenter un algorithme par une suite $A=(a_i)_{i=0..T}$ $T\in$ IN où a_i est une application de $E^{v(i)}$ dans $E^{v(i+1)}$ et où $v(i)\in$ IN désigne le nombre de variables mémorisées à l'étape i.

En algorithmique numérique, on manipule en fait deux grandes classes d'algorithmes, les **algorithmes itératifs** qui correspondent au cas **T infini** et les **algorithmes finis** qui sont associés à **T fini**. Nous nous intéressons à l'exécution de ces algorithmes pour des données particulières. Dans le cas où T est fini, cela ne pose aucun problème, l'exécution A(d) de a pour la donnée d est obtenue par le déroulement de la dynamique

$$X_{t+1}=a_t(X_t)\ t=0..T-1,\ X_0=d.$$

Si T est infini, par contre, l'exécution en d n'a de sens que si pour $d \in E^{v(0)}$ $A^t(d)=((a_i)_{i=0..t-1})(d)$ a une limite lorsque t tend vers l'infini. Le terme "limite" étant pris au sens très large, ensembliste, celui d'attracteur.

En pratique, pour résoudre un problème complexe A(d) sur un ensemble de données D on considère aussi souvent des algorithmes approchés, c'est à dire qu'on se donne une **suite d'algorithmes** (A_h) **convergeant** vers un "algorithme solution" (c'est à dire telle que pour tout d de D, $\lim_{h\to 0}(A_h(d)) = A(d)$).

De plus, lors de la résolution complète du problème sur une machine par un algorithme A sur le corps des réels (l'algorithme "théorique"), on observe son **image machine** dans S $\tau(A)=(\tau(a_l))_{l=0..T-1}$. On voudrait que cette représentation soit fidèle.

A-2-2 Simulation des algorithmes.

Tout d'abord, le praticien souhaiterait que les résultats produits par sa machine soient quantitativement bons c'est à dire que pour un algorithme A et pour toute donnée d, $i\{\tau(A)[\tau(d]\} - A(d)$ soit faible.

Mais au delà de ce désir légitime, les concepteurs de logiciels voudraient que les arguments théoriques utilisés lors de l'analyse de leurs algorithmes en supposant l'arithmétique exacte, en particulier en ce qui concerne les approximations numériques et les convergences restent valables en pratique. On aimerait ainsi pouvoir imposer :

• Pour tout d, $A(d) = B(d) \Rightarrow \tau(A)$ proche de $\tau(B)$.
• $(A_h) \rightarrow_{h \rightarrow 0} A \Rightarrow (\tau(A_h))$ tend vers $\tau(A)$ pour $h \rightarrow 0$, ou au moins vers un attracteur inclus dans un voisinage de $\tau(A)$ (pas de dénaturation machine de la convergence réelle).

Enfin, on aimerait pouvoir visualiser expérimentalement des phénomènes trop complexes à analyser théoriquement tels que des convergences en les simulant sur ordinateur. C'est pour cela qu'on aimerait aussi avoir par exemple :
• $(\tau(A_t))_{t \in \mathbb{N}}$ converge $\Rightarrow (A_t)_{t \in \mathbb{N}}$ converge. La convergence étant là aussi prise en un sens large (celui de convergence vers un attracteur).

C'est la réalisation plus où moins parfaite de l'ensemble de ces désirs par une arithmétique qui nous semble devoir constituer le "cahier des charges" d'une bonne implantation.

A-3 Les arithmétiques usuelles

Ordinairement S est inclus dans $\mathbb{R}$ et le processus de représentation est incarné par un arrondi $\Delta : \mathbb{R} \rightarrow S$. Kulisch et Miranker ont donné dans [KM1] et [KM2] une définition de l'arrondi qui permet de garantir que la représentation choisie conserve la relation d'ordre sur les réels et approche tout nombre par un des éléments de S l'encadrant.

Définition 1.
$\Delta : \mathbb{R} \rightarrow S$ est un arrondi si et seulement si :
$\forall x \in S, \Delta x = x$
$\forall (x_1, x_2) \in \mathbb{R}^2, x_1 \leq x_2 \Rightarrow \Delta x_1 \leq \Delta x_2$

On choisit alors pour la représentation τ des réels (un arrondi Δ étant donné):

$$\forall x \in \mathbb{R}, \ \tau(x) = \Delta x$$
$$i = \tau^{-1}$$

Une telle représentation des réels a évidemment des propriétés particulières [Pic]. On a par exemple :

- $\forall X \in S, \tau^{-1}(X)$ est un intervalle.
- $\forall x \in \mathbb{R}, \tau(x) \in \{ \sup \{ y \in S \mid y \leq x \}, \inf \{ y \in S \mid y > x \} \}$.

Ce qui nous permet ainsi de définir la représentation des réels dans une arithmétique usuelle par la donnée d'une famille $(I_{xi}, x_i)_{i \in J} \in \mathfrak{I}_{\mathbb{R}} \times \mathbb{R}$ telle que pour tout indice i de J :

- $x_i \in I_i$
- $\tau / I_{x_i} = x_i$

(on désigne par $\mathfrak{I}_{\mathbb{R}}$ l'ensemble des intervalles de $\mathbb{R}$)
La plupart des auteurs [KM1], [Pic], [Bro], [Yoh] proposent alors de se donner une famille d'opérateurs machine élémentaires $O = (F_i)_{i=1..n}$ représentant des opérateurs élémentaires $(f_i)_{i=1..n}$ dits canoniques dans S muni de Δ, de la manière algébriquement la plus correcte, c'est à dire vérifiant:

$$F_i(X_1, \ldots, X_n) = \tau(f_i)(X_1, \ldots, X_n) = \Delta(f(X_1, \ldots, X_n)).$$

On pose alors ensuite $\tau(f_{i_1} o f_{i_2} \ldots o f_{i_p}) = F_{i_1} o F_{i_2} o \ldots o F_{i_p}$ $i_j \in \{1..n\}$, $j = 1 \ldots p$.

Un des intérêts d'avoir des opérateurs élémentaires canoniques et de définir τ par la formule précédente est de pouvoir proposer des méthodes d'estimation de l'erreur sur les résultats d'un calcul sur ordinateur basées sur deux termes génériques (cf chapitre II) :

- L'erreur de représentation des nombres .
- Le conditionnement des f_i, $i=1..n$

Cette exigence de canonicité pour les opérateurs élémentaires est cependant assez restrictive. Elle interdit en particulier de considérer en pratique certaines fonctions de base comme des opérateurs élémentaires [Bro]. C'est pourtant le modèle usuel pris par de nombreux auteurs pour les arithmétiques utilisées. C'est aussi celui que nous retiendrons.

Exemple : les arithmétiques à virgule flottante

Dans la plupart des systèmes informatiques (hormis certains systèmes spécialisés par exemple de calcul formel ou de traitement du signal) les calculs sont effectués avec des arithmétiques à virgule flottante.

Cela correspond à $S = \left\{ \left(\sum_{i=1}^{e} a_i \beta^{-i} \right) \beta^n \mid n \in \{-q, \ldots, +p\}, a_i \in \{0, \ldots, \beta-1\} \; a_1 \in \{1, \ldots, \beta-1\} \right\}$.

Pour $X = \left(\sum_{i=1}^{e} a_i \beta^{-i} \right) \beta^n = m.\beta^n$; m est nommée mantisse de X et n exposant de X. On notera par la suite exp(X) l'exposant de X.

Dans la pratique la base β est égale à deux, à une puissance de deux ou à dix (cas des calculatrices). Les arrondis usuels utilisés, la troncature, l'arrondi au plus près, l'arrondi par excès, l'arrondi par défaut et l'arrondi vers l'infini sont définis ci dessous pour z réel :

- Arrondi par défaut $Z = \nabla(z) = \text{Max}\{Y \in S \mid Y \leq z\}$.
- Arrondi par excès $Z = \Delta(z) = \text{Min}\{Y \in S \mid Y \geq z\}$.
- Arrondi au plus près $\forall Y \in S, |z - Y| \geq |z - Z|$.
- Troncature ou arrondi vers zéro $Z = \nabla(z)$ si $z \geq 0$, $\Delta(z)$ si $z < 0$.
- Arrondi vers l'infini $Z = \Delta(z)$ si $z \geq 0$, $\nabla(z)$ si $z < 0$.

Les opérateurs élémentaires canoniques choisis quant à eux sont la plupart du temps ceux correspondant à la structure de corps de $\mathbb{R}$; c'est à dire les quatres opérateurs arithmétiques d'addition de soustraction de multiplication et de division. Les opérateurs machine élémentaires associés seront notés par la suite par entouré. L'implantation machine de l'addition sera par exemple représentée par $\oplus$. Puisque c'est le cas le plus rencontré en pratique, l'arithmétique flottante nous servira pour illustrer les problèmes posés par une arithmétique discrète.

B - DEFAUTS DES ARITHMETIQUES USUELLES.

B-1 Altération de la valeur des résultats.

B-1-1 Singularisations.

En arithmétique usuelle, autour de chaque nombre machine X, il existe un intervalle I_X dont tous les éléments sont arrondis en X. L'arrondi singularise donc les réels par intervalles. Lors d'un arrondi unique non suivi de calculs, l'information perdue est négligeable devant le résultat obtenu. En se produisant itérativement ou en étant suivie d'un opérateur mal conditionné cette singularisation peut par contre modifier de manière plus significative les résultats.

•***Exemple d' "absorptions" successives :***

Ainsi en arithmétique flottante, par exemple, lors de l'addition de deux nombres x et y positifs d'exposants respectifs exp(x) et exp(y) et tels que x est supérieur à y, on perd les exp(x)-exp(y) chiffres de droite de y. Ainsi en troncature sur n chiffres, si $A=\beta^r$, $A \oplus [0. .\beta^{r-n-1}] = \{A\}$. Nous conviendrons d'appeler cette singularisation lors de l'addition *absorption* .

Là encore, une seule absorption est sans grande conséquence. Par contre, il est possible que lors d'une longue sommation, l'erreur produite par de telles absorptions ne soit plus du tout négligeable devant le résultat final. L'exemple qui suit n'est qu'un "exemple d'école", toutefois le phénomène qu'il décrit se produit presque chaque fois que l'on doit additionner un grand nombre de termes (par exemple, lors d'une intégration numérique).
On cherche à calculer le plus précisément possible, en simple précision sur SUN 3 :

$$\sum_{i=1}^{n} \frac{1}{i}$$

pour n grand.

Bien entendu, ces termes peuvent être ajoutés dans n'importe quel ordre. La première idée qui vient à l'esprit est d'effectuer cette addition dans l'ordre croissant des indices, qui est l'ordre naturel de la série, c'est à dire d'effectuer l'algorithme :

pour i = 1 jusqu'à n faire somme := somme + 1/i

il est toutefois bien connu qu'il est en général préférable d'effectuer une addition en commençant par les petits termes, afin d'éviter que ceux-ci ne soient rendus négligeables devant la variable qui représente les termes déjà additionnés, et qu'il y ait par conséquent un problème d'absorption. On peut donc préférer effectuer l'algorithme suivant :

pour i = n jusqu'à 1 faire somme := somme + 1/i

La table suivante présente les valeurs calculées de cette série harmonique pour n = 10 000, 1 000 000, 10 000 000 et 100 000 000, ainsi que les valeurs exactes. On constate effectivement que la sommation effectuée en commençant par les petits termes donne de bien meilleurs résultats, surtout dans les cas n = 10 000 000 et n = 100 000 000, que la sommation dans l'ordre naturel (pour laquelle on n'a qu'un seul chiffre significatif).

n	10000	1 000 000	10 000 000	100 000 000
indices croissants	9.787613	14.35736	15.40368	15.40368
indices décroissants	9.787604	14.39265	16.68603	18.80792
valeur exacte	9.787606	14.39273	16.69531	18.99789

•***Exemple de destruction :***

Un autre exemple où la singularisation peut se révéler catastrophique est le phénomène de destruction (encore appelée cancellation). Ce phénomène se produit lorsque l'on soustrait deux quantités de valeurs proches et constitue le principal révélateur d'erreur en calcul numérique (on utilise à dessein le terme "révélateur" parce qu'en fait, lors d'une cancellation, la soustraction est effectuée exactement : l'erreur qui apparait est la conséquence d'erreurs antérieures, qui sont alors *révélées* [Knu]).

Supposons par exemple que l'on travaille en base 10, avec des mantisses de quatre chiffres et un arrondi au plus près, et que l'on effectue la soustraction C = A - B, où A et B sont les deux nombres machine :

$$A = 0.4205 \times 10^{1} \qquad B = 0.4199 \times 10^{1}$$

le résultat obtenu est :

$$C = 0.6000 \times 10^{-2}$$

Les trois derniers zéros de la mantisse de ce résultat sont complètement artificiels, et C n'a qu'un seul chiffre significatif, qui est le "6". En effet :

• Le nombre machine A peut parfaitement représenter le réel 4.205486, et B peut représenter 4.198787, auquel cas, C devrait valoir 0.6699×10^{-2}.

• Mais si A représente le réel 4.204786 tandis que B représente 4.199432, C devrait valoir 0.5354×10^{-2}.

Le seul moyen d'éviter ce phénomène est de réécrire ses algorithmes et ses programmes de manière à éviter de soustraire des quantités de même ordre de grandeur... ce qui est plus facile à dire qu'à faire. On pourra à ce sujet consulter les travaux de La Porte [Por] donnant une formule numériquement stable (sans soustraction catastrophique) de résolution d'une équation du 3ième degré. On peut tout de même recommander aux utilisateurs d'effectuer des changements de variables lors de l'écriture des programmes de manière à placer les origines des grandeurs physiques au voisinage des valeurs utilisées.

Exemples.

• On veut approcher une dérivée $\partial f/\partial t$ par un taux d'accroissement $\frac{f_2-f_1}{t_2-t_1}$. Le temps t_1 est le 14 Juillet 1989 à 10h52mn35.216436sec., le temps t_2 est le 14 Juillet 1989 à 10h54mn26.327651sec. Si l'on prend comme origine des temps le 0 Janvier 1900 à 0 heure, on a :

- $t_1 = 62743805555.216436$
- $t_2 = 62743805666.327651$
- $t_2 - t_1 = 111.111215$

Si on calcule sur SUN3 (arithmétique IEEE) en simple précision, on obtient $t_2 - t_1 = 0$ sec. : il n'y a plus aucun chiffre significatif, car toute la précision du système à été consacrée à la représentation d'une information inutile (14 Juillet 1989 à 10 heures...).

Si par contre on prend comme origine des temps le 14 Juillet 1989 à 10h50mn, on obtient :

- $t_1 = 155.216436$
- $t_2 = 266.327651$

et le résultat du calcul en simple précision est $t_2 - t_1 = 111.1112$ sec. : tous les chiffres sont significatifs. Cet exemple peut sembler un peu exagéré, mais nombre de programmes de calcul scientifique sont remplis de ce genre de problème.

• Un exemple plus amusant des conséquences du phénomène de destruction est cité par W.J. Cody dans un article consacré à la précision du calcul de fonctions : il s'agit de calculer le sinus de 22. Les méthodes de calcul des fonctions trigonométriques (approximations polynomiales ou rationnelles, algorithme CORDIC [Mul], ...) ne convergent que sur un petit intervalle I (généralement $[-\pi/2, \pi/2]$). Lorsque l'on veut calculer le sinus d'un nombre x n'appartenant pas à cet intervalle, on cherche tout d'abord, par exemple en effectuant une division, un entier N et un réel $y \in I$ tels que y soit égal à $x - N\pi$, puis on calcule sin (y). On obtient enfin : $\sin(x) = (-1)^N \sin(y)$. Lorsque x est très proche d'un multiple de π, ce qui est le cas pour $x = 22$ puisque 22/7 est une excellente valeur approchée de π, $N\pi$ est très proche de x, et par conséquent, le calcul :

$$y = x - N\pi$$

produit un phénomène de destruction.

Heureusement, il existe des algorithmes de réduction d'argument plus sophistiqués qui permettent de limiter ce problème, et qui reviennent à simuler une plus grande précision. Ces algorithmes ne sont hélas pas toujours implantés, comme le montre la table suivante :

machine	$-10^3 \sin(22)$
valeur exacte	8.8513 09290 40388
SUN 3 (avec ou sans coprocess.)	8.8513 09290 40388
Micro Vax	8.8513 09290 40388
TI59	8.8513 09285 516
Casio FX702P	8.8513 09219
Sharp PC1245	8.8513 09503
Sharp EL512	8.8513 09503
IBM PC AT (Basic double prec.)	8.8513 09306 91957
HP65	8.8513 0632
TI25	8.8487

Les pertes d'information, par exemple par absorption ou destruction, peuvent donc conduire à des résultats catastrophiques : le résultat calculé peut même parfois différer de plusieurs ordres de grandeur du résultat correct. Considérons pour conclure ce paragraphe un dernier problème. Les calculs ont été menés à bien en utilisant la bibliothèque SANE d'Apple, en précision simple et double. On cherche à calculer la somme des nombres a = 1, b = 2.0e+22, c = -2.0e+8, d=1.0e+22, e = 2.0e+8, f = -9.0e+21, g = -1.0e+21. Le résultat correct est 1. On obtient, suivant l'ordre dans lequel est effectuée la sommation :

ordre de sommation	simple précision	double précision
a+b+c+d+e+f+g	-4.925812e+14	0.000000000000000e+0
b+d+f+g+c+e+a	-4.925812e+14	1.000000000000000e+0
a+c+b+d+f+g+e	-4.925812e+14	5.120000000000000e+2
c+f+g+b+d+e+a	-4.925812e+14	5.130000000000000e+2
a+c+b+d+e+g+f	-4.925812e+14	0.000000000000000e+0
g+f+e+d+c+b+a	-4.925812e+14	1.000000000000000e+0

On observe dans certains cas une perte d'information "totale". Que s'est-il produit ?

Considérons par exemple le cas de la première ligne : en simple précision, les termes a, c et e ont été absorbés (ce sera d'ailleurs le cas de tous les autres calculs en simple précision), donc le seul calcul qui ait été effectivement réalisé est b+d+f+g. Le résultat de ce calcul aurait dû être zéro, mais comme b, d, f et g ne peuvent s'écrire exactement en simple précision, ce calcul a été effectué de manière approchée, et on a donc commis une erreur de l'ordre de grandeur du poids du dernier bit de la représentation en simple précision de 2.10^{22}. L'exposant de cette représentation est 75, puisque 2^{22} vaut 0.52939... 2^{75}, donc le poids du dernier bit est $2^{75-24} \approx 2.10^{15}$.
En double précision, a, c et e ont également été absorbés, et on a calculé b+d+f+g = 0.

A la deuxième ligne, en double précision, on a tout d'abord calculé b+d+f+g=0, par conséquent, les termes a, c et e n'ont pas été absorbés, et on a obtenu le résultat correct.

B-1-2 Délocalisation.

Un phénomène presque "dual" de celui de singularisation est le phénomène de délocalisation : certaines pertes d'information dues aux arrondis peuvent être vues comme des effets de dispersion [PV].

Soit f un opérateur continu et F une représentation machine de f ($F= \tau((f_i)_{i=0..T})$ avec $((f_i)_{i=0..T})=f$). Pour que l'approximation discret-continu soit correcte en un nombre machine x, F(x) doit représenter $f(I_X)$ où I_X est l'intervalle machine contenant x. Comme f est continue $f(\underline{I_X})$ est égal à $\underline{f(I_X)}$ ($\underline{O}$ désigne l'adhérence de O).

Pour que F soit une bonne approximation de f, il faut donc que $F(I_X \cap S)=F(S_X)$ représente bien f(x). Plus généralement, pour un algorithme de calcul de $f(x_0)$ de la forme $x_{j+1} := f_j(x_j)$ $j=0...T$ où les f_j sont des opérateurs canoniques, on imposera à l'algorithme machine correspondant : $X_{j+1}=F_j(X_j)$, $X_0 = \Delta(x_0)$ que la suite de nuages de points récurrente :

$$N_0 := I_{X_0} \cap S.$$

$$N_{j+1} := \cup_{x \in N_j} (I_{F_j(x)} \cap S).$$

soit telle que N_{T+1} représente bien $f(x_0)$ au sens intuitif du terme.

Le problème est que pour certains algorithmes, le nuage de points peut être tellement dispersé qu'aucune information ne peut être extraite de la population finale $N(F(X_0))$. Dans ce cas $f(x_0)$ est *délocalisé* par F. Approcher $f(x_0)$ par $F(X_0)$ n'a alors pas *a priori* de signification.

Cas important.

Si un algorithme s'écrit "si $f(x)<0$ alors $p_1(x)$ sinon $p_2(x)$ " avec $p1 \neq p2$ sur la variété $f(x)=0$, il est intéressant de détecter de telles "singularités arithmétiques" de l'image F de f. En effet, si F est trop délocalisante au voisinage d'un point x tel que $f(x)=0$, on ne sait pas a priori déterminer en machine si $f(x) >0$.

B-2 Déstructurations.

Au delà de l'incapacité de l'arithmétique machine à donner de bonnes valeurs numériques, et peut-être plus gravement, les arrondis détruisent aussi parfois la morphologie même des problèmes traités: les résultats calculés peuvent différer des vrais résultats non seulement quantitativement mais aussi qualitativement.

B -2-1 Déstructuration topologique.

Un premier exemple est d'ordre topologique. Le comportement (convergence, points fixes ...) d'une itération I dans $\mathbb{R}^n$ et celui de l'itération machine $\tau(I)$ qui lui est associée peuvent être complètement différents. Tous les cas de figures défavorables semblent pouvoir se produire.

• F ROBERT [Rob] a mis en évidence des phénomènes de duplication de points fixes ou d'apparition de cycles dans des itérations affines. P BINDER [Bin] vient récemment de les expliciter complètement en arithmétique virgule fixe.

• A l'inverse, on a également des exemples où la suite reste globalement convergente par passage sur machine mais où le nombre de limites machine est inférieur au nombre réel de limites.

Ainsi si $x_{n+1} = f(x_n)$ est une suite convergente vers L pour $x_0 \in \mathbb{R}^n - A$ et vers M pour $x_0 \in A$, où A est un ensemble de mesure de Lebesgue nulle alors généralement $X_{n+1} = F(X_n)$ convergera toujours vers L. C'est le cas de la suite récurrente donnée en introduction.

• Enfin, on peut aussi mettre en évidence l'apparition de convergences ou de divergences non prévues à cause de phénomènes de singularisation.

-Toute série positive de terme général tendant vers 0 est par exemple convergente en machine .

- La suite :

$$u_n = ((((5^n - 5^{n-1}) - 5^{n-1}) - 5^{n-1}) - 5^{n-1}) - 5^{n-1}$$

stationnaire (et donc convergente) en réalité est divergente en machine, comme l'illustre la table suivante des valeurs de u_n calculées en double précision sur SUN3.

n	*valeur calculée de* u_n
de 1 à 22	0
23	-1
24	4
25	-32
26, 27	0
28	3072
29	-8192
30	-32768
40	0
41	$1.099511...10^{16}$
50	0
60	$3.868...10^{25}$
70, 80, 90	0
100	$1.9156...10^{53}$
200	0
300	$-1.82497...10^{193}$

Divergence de la suite u_n

B-2-2 Déstructuration algébrique.

Un deuxième type de différence qualitative est d'ordre algébrique. Ainsi l'addition virgule flottante n'est pas associative. Par exemple, si on travaille en arithmétique virgule flottante en base 10, sur quinze chiffres, avec la troncature :

$$10^{22} \oplus (-10^{22} \oplus 10^{-15}) = 0 \neq (10^{22} \ominus 10^{22}) \oplus 10^{-15} = 10^{-15}.$$

La structure de corps des réels n'est donc pas préservée. Plus généralement, comme nous l'avons vu précédemment (dans le cas de la série harmonique ou du calcul de la somme de six nombres a...g), les images informatiques de deux algorithmes équivalents c'est à dire produisant théoriquement les mêmes valeurs ne sont pas toujours équivalentes. Il se pose alors la question de trouver des transformations d'algorithmes permettant d'améliorer la qualité de l'image informatique. On peut aussi se demander de quelle structure algébrique est muni S [Cla].

B-2-3 Une déstructuration "exotique".

Les exemples donnés précédemment sont pour la plupart assez classiques. Un exemple plus original de déstructuration qualitative que nous avons mis en évidence au cours de notre étude est maintenant exposé. On considère la famille de polynômes de degré deux $P(\varepsilon_1,\varepsilon_2)= x^2+(a+\varepsilon_1)x+(b+\varepsilon_2) = 0$ avec $a^2 = 4b$ et ε_i, $i=1..2$ variables à support sur $[-\alpha_i,\alpha_i]$ de densité strictement positive .
On s'intéresse alors à l'ensemble $\mathcal{R}(\alpha_1,\alpha_2)$ des racines $\{x_j(\varepsilon_1,\varepsilon_2), j=1..2\}$ de cette famille.
Le théorème suivant explicite sa structure :

Théorème.
Si $\alpha_1<|a|$ et $\alpha_2<|a|/2$, $R(\alpha_1,\alpha_2)$ est homéomorphe suivant les valeurs respectives de α_1 et α_2 à l'un des trois ensembles dont la frontière est dessinée ci dessous.

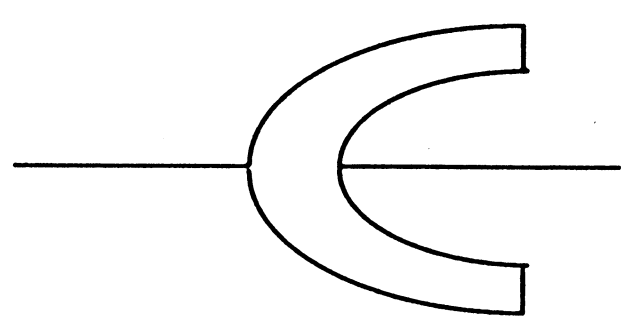


figure 1 : cas $\alpha_1>+|a|-\sqrt{a^2-4\alpha_2}$

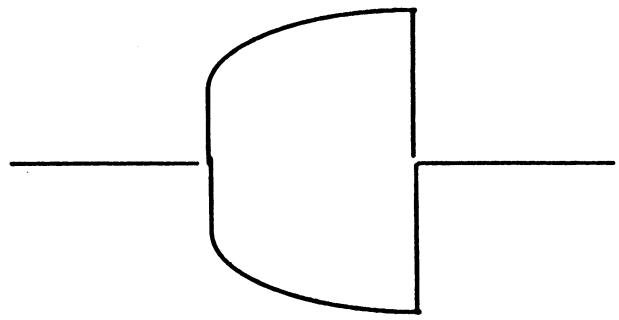


figure 2 : cas $\alpha_1 < -|a| + \sqrt{a^2 + 4\alpha_2}$

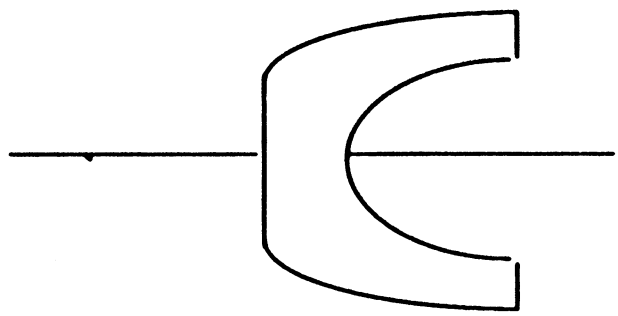


figure 3 : cas $-|a| + \sqrt{a^2 + 4\alpha_2} < \alpha_1 < +|a| - \sqrt{a^2 - 4\alpha_2}$

Démonstration.

Les racines de $P(\varepsilon_1, \varepsilon_2)$ s'écrivent :

$$x_1(\varepsilon_1, \varepsilon_2) = \frac{-a - \varepsilon_1 + \sqrt{\varepsilon_1^2 + 2a\varepsilon_1 - 4\varepsilon_2}}{2}$$

$$x_2(\varepsilon_1, \varepsilon_2) = \frac{-a - \varepsilon_1 - \sqrt{\varepsilon_1^2 + 2a\varepsilon_1 - 4\varepsilon_2}}{2}$$

Supposons $a > 0$ et étudions l'ensemble $\mathfrak{R}$ qu'elles décrivent.

Pour ε_1 inférieur à $-|a| + \sqrt{a^2 - 4\alpha_2}$ on obtient comme racines (en faisant varier ε_2) tous les complexes de partie réelle $-(a+\varepsilon_1)/2$ et de partie imaginaire de valeur absolue comprise entre $\sqrt{-(\varepsilon_1^2 + 2a\varepsilon_1 - 4\alpha_2)}$ et $\sqrt{-(\varepsilon_1^2 + 2a\varepsilon_1 + 4\alpha_2)}$. Cela justifie la partie de $\mathfrak{R}$ comprise entre les 2 branches d'ellipses dans le cas où $\alpha_1 > +|a| - \sqrt{a^2 - 4\alpha_2}$ (figures 1 et 3).

Pour ε_1 compris entre $-|a| + \sqrt{a^2 - 4\alpha_2}$ et $-a + \sqrt{a^2 + 4\alpha_2}$ on obtient cette fois tous les nombres complexes de partie réelle $-(a+\varepsilon_1)/2$ dont la partie imaginaire a une valeur absolue inférieure à $\sqrt{-(\varepsilon_1^2 + 2a\varepsilon_1 - 4\alpha_2)}$ ainsi que des réels purs. Cela justifie une partie de $\mathfrak{R}$ entre l'ellipse supérieure et l'axe horizontal (figures 1,2 et 3).

Enfin, pour ε_1 supérieur à $-a + \sqrt{a^2 + 4\alpha_2}$, on a par contre uniquement des racines réelles. D'où une branche réelle pour $\mathfrak{R}$ comme celle à gauche de la figure dans ce cas (figure 1).

Nous avons essayé de visualiser $\mathfrak{R}$ en machine. Pour cela, nous avons calculé les racines d'un polynôme P par la méthode de Durand Kerner, c'est à dire par l'itération

$$(x_1^0, x_2^0) \text{ complexes}$$

$$x_1^{n+1} = x_1^n - \frac{p(x_1^n)}{x_1^n - x_2^n}$$

$$x_2^{n+1} = x_2^n - \frac{p(x_2^n)}{x_2^n - x_1^n}$$

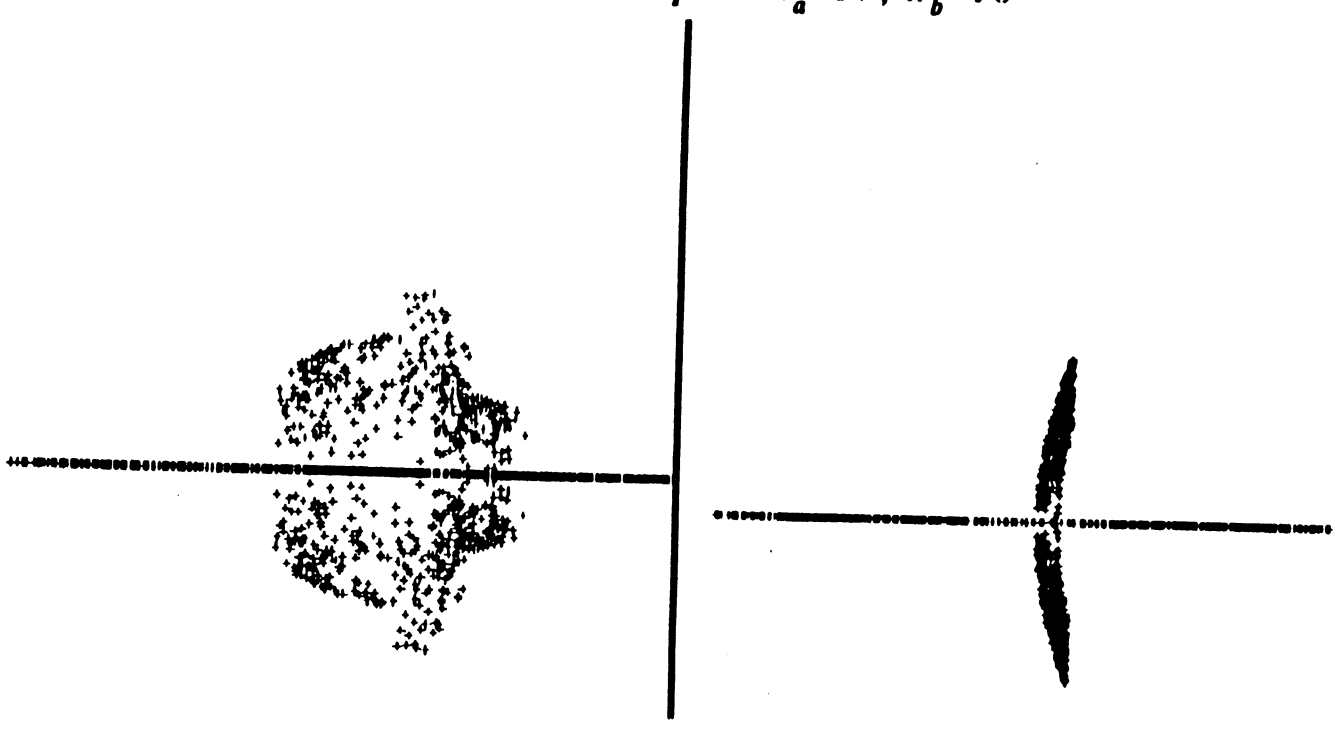
En théorie, d'après le théorème précédent l'ensemble $E_n = \{x_j^n(\varepsilon_1, \varepsilon_2)\ j=1..2\}$ devrait converger quand n tend vers l'infini vers les surfaces décrites dans le théorème précédent (suivant les structures de perturbations choisies). Nous avons pris différents polynômes et appliqué à ces polynômes des perturbations uniformes aux derniers bits au niveau des coefficients; c'est à dire des perturbations correspondant à :

$\alpha_1 = 2^{\exp(a)+n_a-L}$, $\alpha_2 = 2^{\exp(b)+n_b-L}$ où L est la longueur de la mantisse des nombres machine.

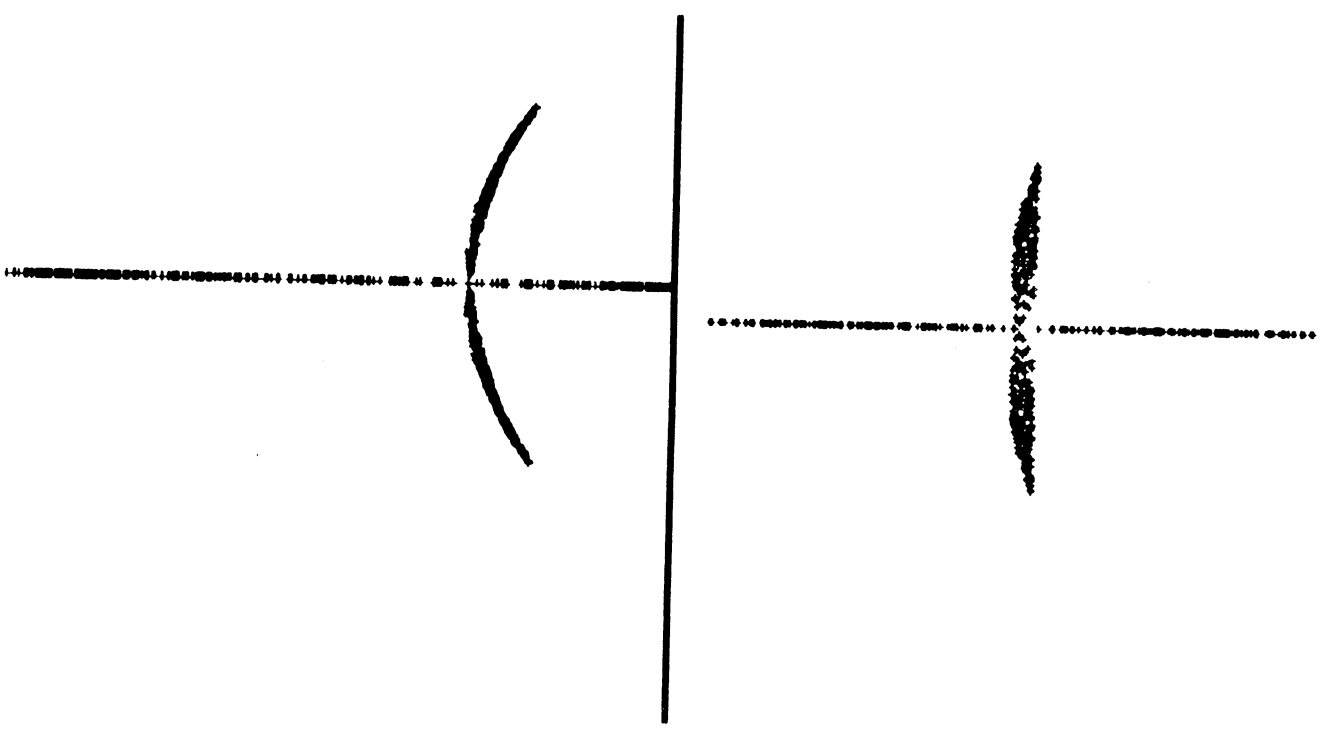
Exemple :

Les résultats sont visualisés graphiquement ci dessous pour a=2 sur 1000 échantillons pour deux valeurs différentes du couple (n_a, n_b). Les points de départ ont étés pris égaux respectivement à 2+2i et 1+i et on a effectué 50 itérations. On s'est assuré que la convergence numérique était atteinte pour ce nombre d'itérations en visualisant qu'à partir de ce nombre d'itération la morphologie du nuage était inchangée (tout au moins jusqu'à 10000 itérations). On visualise à droite la figure théorique obtenue à l'aide de la formule du discriminant et à gauche la figure obtenue par la méthode de Durand Kerner.

Destructuration pour $n_a=17$, $n_b=18$



Destructuration pour n_a=18, n_b=17



On observe donc une déstructuration, due aux erreurs d'arrondis, de la nature géométrique de l'ensemble limite des E_n correspondant à α_1 et α_2 donnés. Il semble par contre que l'ensemble des ensembles limites pour α_1 et α_2 variables soit lui invariant par perturbation. Tout se passe comme si lorsqu'on calcule $\mathfrak{R}(\alpha_1,\alpha_2)$ en machine par la méthode de Durand Kerner, on calculait en fait une partie de $\mathfrak{R}(\alpha'_1,\alpha'_2)$ avec $\alpha'_1>\alpha_1$ et $\alpha'_2>\alpha_2$. Ceci peut s'interpréter en voyant les erreurs d'arrondi dans la méthode de Durand Kerner comme une perturbation des coefficients du polynôme dont on cherche les racines; c'est à dire en effectuant ce que l'on nomme une analyse rétrograde des perturbations dues aux arrondis (cf chapitre III)

B-3 Conclusion.

L'utilisation d'arithmétiques usuelles peut donc poser de graves problèmes de non représentativité quantitative des résultats, ou de destruction de leur structure mathématique.

Certaines des différences entre un algorithme mathématique et sa représentation machine sont inévitables parce qu'intrinsèquement liées au problème. C'est le cas du calcul de fonctions discontinues. Il est cependant intéressant d'essayer de choisir les arithmétiques de manière, d'une part à limiter les problèmes, d'autre part à détecter leur occurrence.

C – CHOIX DES ARITHMETIQUES.

C – 1 Optimisation des arithmétiques usuelles.

On peut d'abord se limiter aux arithmétiques usuelles et tenter de rationnaliser le choix de certains paramètres (nombres représentables, opérateurs élémentaires canoniques).

C-1-1 Choix de S et de Δ.

C-1-1-1 Quels principes ?

a) Principes de réalisabilité.

Les premières contraintes des concepteurs d'arithmétiques machine sont des contraintes matérielles et de temps de calcul [Mul]. Les nombres utilisés doivent pouvoir être représentés dans des cellules mémoire de taille fixée et les opérations élémentaires canoniques doivent être simples à réaliser et s'effectuer rapidement. C'est essentiellement à ces contraintes qu'est consacrée la majorité de la littérature spécialisée en arithmétique machine.

b) Principe quantitatif.

Cependant, à la lumière des problèmes présentés précédemment, le deuxième type de difficulté sur laquelle se penchent les arithméticiens machine est la contrainte de représentation quantitative des algorithmes. On cherche, à taille mémoire fixée et à opérateurs choisis, à minimiser l'erreur de représentation. Les publications se limitent généralement à l'erreur de représentation des nombres (étude de l'erreur statique) en espérant ou en justifiant empiriquement la persistance des résultats lors de calculs (voir à ce propos le chapitre suivant). Différents critères de mesure de l'erreur de représentation sont retenus [Bre]. Ils peuvent être déterministes (mesure du pire des cas : norme L_∞) ou probabilistes (erreur quadratique moyenne, sous certaines hypothèses sur la distribution des nombres rencontrés), relatifs ou absolus. Cependant, la plupart des auteurs choisissent comme critère l'erreur relative quadratique moyenne entre x et $\tau(x)$, soit $\left[\int \left|\frac{x - \tau(x)}{x}\right|^2 \mu(x)\, dx\right]^{\frac{1}{2}}$ où $\mu(x)$ est la densité choisie pour x, et supposent en virgule flottante une répartition logarithmique des mantisses :

$$\text{probabilité (mantisse} \leq m) = \frac{\log m + \log b}{\log b}$$

où b est la base de la représentation choisie ([Bre],[Knu],[SS]). Nous évoquerons plus en détail ces modèles au chapitre suivant à l'occasion de la présentation des modèles probabilistes globaux de l'erreur d'arrondi.

c) Principes algébriques.

Mais au delà de cette optimisation quantitative, il peut aussi sembler intéressant d'imposer à $\tau = (S,\Delta)$ de vérifier des propriétés plus qualitatives, par exemple de représentation de la structure de corps des réels. Nous avons ainsi cherché à spécifier concrètement (en terme de propriétés de symétrie) les systèmes de nombres vérifiant certaines des contraintes suivantes :

$P_1)\ \forall X \in S, \forall y \in I_\tau(0), \tau(X+y)=X.$	(neutralité des zéros machine pour l'addition)
$P_2)\ \forall x \in \mathbb{R}, \tau(\tau(x)+\tau(-x))=\tau(0)$	(-x est l'opposé de x à la précision machine)
$P_3)\ \forall X \in S, \forall y \in I_\tau(1)\ \tau(X * y)=X.$	(neutralité des 1 machine pour la multiplication)
$P_4)\ \forall x \in \mathbb{R}, \tau(x)\neq 0 \tau(\tau(x) * \tau(1/x))=\tau(1)$	(1/x est l'inverse de x à la précision machine)

Cela a débouché sur les résultats suivants

d) Deux Théorèmes.

Définition préliminaire.

On dira que x est un zéro machine si et seulement si $\tau(x) = \tau(0)$.

De la même manière, on dira que x est un 1 machine si et seulement si $\tau(x) = \tau(1)$.

Théorème relatif à l'addition.

Les conditions :

$P_1)\ \forall X \in S, \forall y \in I_\tau(0), \tau(X+y) = X.$

$P_2)\ \forall x \in \mathbb{R}, \tau(\tau(x)+\tau(-x)) = \tau(0).$

sont vérifiées si et seulement si :

i) $\tau(0) = 0$

ii) $\forall X \in S,$

a) si x-X est un zéro machine alors $\tau(x) = X$

b) $I_\tau(-X)=-I_X$

c) $X+\tau(-X)$ est un zéro machine.

Démonstration.

On démontre d'abord le lemme suivant :

Lemme préliminaire :

P1) $\Leftrightarrow$ i) $\tau(0)=0$

ii)-a) si x-X est un zéro machine alors $\tau(x) = X$.

Démonstration du lemme.

Si P1 est vraie, alors pour tout $X \in S$, pour tout réel x tel que x-X soit un zéro machine, $\tau(x)$ est égal à $\tau(X+x-X)=X$. Par suite, ii)-a) est vraie. Par ailleurs, si on avait $\tau(0) \neq 0$, on aurait $\tau(\tau(0)+\sup_{y \in I_{\tau(0)}}(y)) = \tau(0)$ et donc : $\tau(0)+\sup_{y \in I_{\tau(0)}}(y) \in I_{\tau(0)}$. Par suite, $\tau(0) = 0$. i) est donc également vraie.
La réciproque est immédiate. En effet, si ii-a) est vraie, alors, pour $y \in I_{\tau(0)}$ et X de S, (X+y)-X est un zéro machine et donc $\tau(X+y)=X$.

Démonstration du théorème.
Partie $\Rightarrow$
Nous remarquerons d'abord que, par définition, P2) implique ii)c), il suffit donc de montrer

$$(P1), i) \text{ et } ii)\text{-a) et P2)) } \Rightarrow \text{ ii) b)}$$

Supposons donc i) et ii)a), P2) et P1) vérifiées.
Alors, pour x dans $I_{\tau(0)}=I_0$, on a :

$$\tau(-x)=\tau(\tau(-x)=0$$

Par conséquent, I_0 est symétrique par rapport à O. Soit maintenant $X \in S$, quelconque, et x dans I_X. On a, d'après P2) :

$$\tau(X+\tau(-x))=0$$

donc $\tau(-x)+-(-X)$ est un zéro machine. Par suite, d'après ii)-a) : $\tau(-x) = \tau(-X)$.
On a donc :

$$\forall X \in S, -I_X \text{ est inclus dans } I_{\tau(-X)} \text{ iii)}$$

par conséquent :

$$\forall X \in S, -I_{\tau(-x))} \text{ est inclus dans } I_{\tau(-\tau(-X))}$$

Par ailleurs, d'après P2), pour tout X de S, $X + \tau(-X)$ est élément de $I_{\tau(0)}$. Par suite, à cause de la symétrie de I_0, $-X-\tau(-X)$ est dans I_0. D'après ii)-a), on a donc :

$$\forall X \in S, \tau(-\tau(-X))=X$$

et donc :

$$\forall X \in S, I_{\tau(-x)} \text{ inclus dans } -I_X$$

ii-b) est donc vérifiée (d'après iii)).

Partie $\Leftarrow$: Elle est aisée. On vérifie en effet facilement que les conditions ii)b) et ii)c) suffisent à satisfaire P2).

D'où le théorème.

Théorème relatif à la multiplication.

Les conditions

P3) $\forall X \in S, \forall y \in I_{\tau(1)}\ \tau(X^* y)=X.$

P4) $\forall x \in \mathbb{R}, \tau(x)\neq 0, \tau(\tau(x)\ ^*\tau(1/x)) = \tau(1)$

sont vérifiées si et seulement si :

i) $\tau(1)=1$

ii) $\forall X \in S-\tau(0)$,

a) si X/z est un 1 machine alors $\tau(z)=X$

b) $X^*\tau(1/X)$ est un 1 machine.

c) $I_{\tau(1/X)}=1/I_X$

Démonstration.
Le principe de démonstration est similaire à celui du théorème sur l'addition.

Remarque : Des propriétés plus complexes comme l'associativité seraient probablement plus difficiles à réaliser.Une réflexion algébrique sur les systèmes de représentation des nombres nous parait cependant intéressante à approfondir.

C-1-1-2 Examen de quelques cas proposés dans la littérature.

Différents systèmes de représentation des nombres sont proposés dans la littérature. Nous passons rapidement en revue les principaux systèmes utilisés [Mul].

a) Arithmétique fixe.

C'est le plus simple et le plus rapide des systèmes. On l'utilise notamment en traitement du signal. Il correspond à $S_f(\beta, e) = \left\{ \pm M^* \Sigma_{i=1...e}\, a_i \beta^{-i} \mid a_i \in \{1...\beta-1\} \right\} \cup \{0\}$. On travaille généralement avec l'arrondi au plus près ou la troncature. Un tel système permet de disposer d'une addition exacte ce qui évite les absorptions. Il pose par contre les mêmes problèmes que l'arithmétique flottante pour les multiplications. De plus, les petits nombres y sont mal représentés, tandis que les grands nombres ne sont pas représentables.

b) Arithmétique logarithmique.

On peut aussi naturellement songer à représenter les nombres par leur logarithme en base a où a est un nombre réel fixé, c'est à dire à travailler dans le système de nombres machine $S_l(a, \beta, e) = \pm a^{S_f(\beta,e)} \cup \{0\}$. Utiliser une telle arithmétique dite "logarithmique" permet de disposer d'une multiplication exacte. Du point de vue du critère d'erreur statique classique elle est également efficace.

L'arithmétique logarithmique sert d'ailleurs souvent de référence [Bre] dans les études sur la virgule flottante. Un tel système de représentation des nombres est aussi structurellement très satisfaisant (par exemple au regard des propriétés algébriques étudiées précédemment). Cependant, l'addition de deux nombres en notation logarithmique est difficile à réaliser. L'implantation proposée par Olver nécessite une addition en virgule fixe et une lecture dans un tableau. Vu la fréquence d'apparition de cette opération dans les calculs scientifiques, c'est un handicap non négligeable.

c) Arithmétique flottante.

C'est le système utilisé par pratiquement toutes les machines. Il réalise un bon compromis entre la simplicité de l'arithmétique fixe et le comportement excellent de l'arithmétique logarithmique. Bien que la structure algébrique de la représentation logarithmique ne soit pas vérifiée localement, elle est assez bien approchée à un niveau plus macroscopique. Du point de vue de l'erreur statique, les travaux de Brent [Bre] montrent que la base 2 avec premier bit implicite est optimale à taille de codage fixé. C'est d'ailleurs le système dans lequel travaillent la plupart des machines scientifiques.

C-1-1-3 Une arithmétique plus exotique.

Certains auteurs ont aussi étudié des systèmes moins connus de représentation des nombres. Olver propose par exemple :

$$S_O = \left\{ \pm e^{\pm(e^{(e^{(\cdots^{e^y})})})} \text{ (n exponentiations)} \mid n \in \mathbb{N} \text{ et } y \in [0,1] \text{ écrit en virgule fixe} \right\}.$$

Ce système présente outre sa double symétrie (si un nombre x s'écrit exactement dans ce système, alors -x et 1/x s'écrivent exactement aussi, et leur écriture se déduit immédiatement de celle de x) l'avantage d'être apte à représenter de très grands et très petits nombres. Cependant, on peut se demander quel est son intérêt pratique. En effet, bien malin qui réalisera un additionneur rapide dans le système d'Olver.

C-1-2 Quels opérateurs canoniques.

Le système de représentation des nombres et le mode d'arrondi d'une arithmétique étant déterminés, il faut aussi choisir des opérateurs canoniques permettant de représenter systématiquement avec la moindre erreur les algorithmes numériques. La plupart des machines à virgule flottante se contentent de fournir à l'utilisateur les opérateurs (+, *, /, -) correspondant à la structure de corps des réels [Mal][Gru]. Une telle approche présente la lacune d'autoriser de fortes erreurs dans les calculs de sommes finies et de produits scalaires, très fréquents (calculs matriciels). Dans le cas de l'arithmétique flottante, Kulisch [KM2] propose d'implanter le produit scalaire comme opérateur canonique. Il montre que cela est réalisable avec un accumulateur de taille double de celle de la représentation des nombres (Voir [Bol][KM1]). Cette solution semble très judicieuse. Elle permet d'implanter un grand nombre d'algorithmes classiques avec une plus grande précision.
En général, on a intérêt à choisir comme opérateurs élémentaires des opérateurs bien conditionnés.

C-1-3 Conclusion.

Des principes de bon sens permettent donc de donner quelques conseils quant au choix des paramètres d'une arithmétique usuelle. Mais, il peut aussi être intéressant de penser l'arithmétique de manière plus large.

C-2 Arithmétiques ensemblistes.

Un des problèmes des arithmétiques usuelles est leur incapacité à rendre compte de la continuité de $\mathbb{R}$ aux "points de rupture" de τ, c'est à dire aux points de discontinuité, inévitables, de τ.

Plus précisément, soit τ associée à $(Ix_i, x_i)_{i\in J}$ telle que (Ix_i) soit une partition de $\mathbb{R}$, $\tau/I_i = x_i$. (on a alors $S = \{x_i\}_{i\in J}$).

Si $x \in I_X^{\circ}$ (I° dénote l'intérieur de I), τ est continue en x, par contre, si $z \in I_X \cap I_Y$, on peut trouver deux suites z_n^1 et z_n^2 tendant vers z telles que :

$$\tau(z_n^1) \to X \text{ et } \tau(z_n^2) \to Y$$
$$n \to +\infty$$

τ n'est donc pas continue en z.

Pouvons nous améliorer τ pour remédier à ce défaut en introduisant une nouvelle représentation des nombres τ'?

Si nous définissons τ' par :

si $z \in S$, $\tau'(z) = z$.
si $z \in I_X \cup I_Y - S$, $\tau'(z) = \{X, Y\}$.
si $z \in I_X^{\circ} - S$ et $I_X = [i_X, s_X]$, $\tau'(z) = \tau'(i_X) \cup \tau'(s_X)$.

alors :

- τ' est continue sur $I_X^{\circ} - S$
- $\forall$ x, $\forall$ $x_n$suite convergeant vers x alors $\forall Z \in \tau'(x)$, $\exists$ $Z_n \in \tau'(x_n)$ tel que Z_n converge vers Z (inf continuité).
- $x \in [\tau'(x)]$, où [A] désigne le plus petit intervalle contenant A.

τ' améliore donc τ du point de vue de la représentation des réels. On peut alors prolonger τ' pour la représentation des algorithmes. Comme $\tau'/\mathbb{R}$ est une application de $\mathbb{R}$ dans $\mathcal{P}(S)$, en posant pour tout f opérateur élémentaire sur $\mathbb{R}^n$ et pour $(\mathcal{X}_i)_{i=1..n}$ éléments de $\mathcal{P}(S)$:

$$F(X_1 \ldots X_n) = \cup_{X_i \in \mathcal{X}_i} \tau'(f(X_1, X_2 \ldots X_n)).$$

on définit, sur le même principe que pour l'arithmétique usuelle, une nouvelle arithmétique "ensembliste". Outre, les propriétés topologiques précédentes, cette nouvelle arithmétique vérifie de plus la propriété suivante :

Propriété.

Si les opérateurs élémentaires sont monotones alors; pour tout algorithme a sur $\mathbb{R}$, $a \in [\tau'(a)]$.
Sinon, pour tout algorithme a sur $\mathbb{R}$ tel que pour tout i

- a_{i+1} est monotone sur $[\tau'(a_i 0 \ldots a_1)]_{i=1\ldots T-1}$
- ou a_{i+1} est monotone sur I_X, $X \in \tau'(a_i 0 \ldots a_1)$ et $\tau'(a_i 0 \ldots a_1))$ est connexe

alors : $a \in [\tau'(a)]$.

où [A] désigne le plus petit intervalle contenant A.

La démonstration est immédiate.

Ainsi donc en passant de l'optique de l'arithmétique usuelle (S partie de $\mathbb{R}$) à l'optique de l'arithmétique ensembliste définie ci dessus, on obtient une arithmétique plus susceptible de représenter les algorithmes réels. Pourtant cette arithmétique pose des problèmes d'implantation et de principe. Tout d'abord parce que la complexité en taille de la représentation d'un algorithme est ici une fonction exponentielle du nombre d'opérations élémentaires de l'algorithme et donc qu'une telle arithmétique n'est pas implantable. Ensuite parce que cette arithmétique ne nous fournit une estimation de l'erreur que si, à chaque étape, on peut vérifier des conditions de monotonicité sur l'opérateur appliqué sur l'ensemble représentant le calcul, ce qui semble de plus en plus difficile au fur et à mesure du calcul (mauvaise simulation de l'image continue d'un connexe par cette arithmétique). Enfin, parce que l'estimation de l'erreur arithmétique obtenue ainsi est par nature majorante.

C-2-1 l'arithmétique d'intervalles.

Une première idée pour éviter le problème de complexité en taille de la représentation est de représenter x directement par [τ'(x)], le plus petit intervalle à bornes dans S contenant τ'(x), c'est à dire de travailler avec des intervalles.

Cela revient à avoir une arithmétique dans S^2. C'est l'arithmétique d'intervalles de Moore [KM2] et elle est simple à formaliser. En posant $\mathbb{J}_S = \{[a,b] \mid a \in S, b \in S\}$ et en définissant un arrondi généralisé $\Delta_{\mathbb{J}}$ de $\mathbb{J}_{\mathbb{R}}$ sur $\mathbb{J}_S$ par $\Delta_{\mathbb{J}}(J) = \inf_{\supset} \{I / I \supset J \mid I \in \mathbb{J}_S\}$, cette arithmétique se ramène à une arithmétique usuelle pour les calculs sur $\mathbb{J}_{\mathbb{R}}$ (on imagine facilement cette notion par analogie à la notion d'arithmétique usuelle dans le cas réel).

Si on souhaite un encadrement certain d'une solution, par exemple dans l'optique de preuve de propriétés mathématiques, cette arithmétique est très intéressante. Cependant, par construction, l'estimation de l'erreur fournie par un tel procédé de calcul ne peut être que pessimiste par rapport à l'erreur effectivement obtenue par un simple calcul dans S.

Ce pessimisme intrinsèque est de plus notablement amplifié par le fait que cette méthode de calcul ne tient pas compte des égalités formelles entre variables [Dav]. Le calcul de l'image de l'intervalle [0,1] par l'expression x^2-x en arithmétique d'intervalle donnera par exemple [-1,+1] alors que la véritable image est [-0. 25, 0].

C-2-2 l'arithmétique stochastique.

Au lieu de représenter x par τ'(x),on peut enfin représenter x par une variable aléatoire τ"(x) à valeurs dans τ'(x). Cela permettra aussi d'éviter le problème de complexité en taille de la représentation. Mais comment choisir l'aléa?

On l'a vu, le problème de l'arithmétique d'intervalles est que la représentation de l'erreur qui en découle est fortement majorante. La première idée est de chercher un aléa minimisant l'erreur de représentation. L'arrondi τ (x) étant fixé, et en se donnant une loi de probabilité $\mathfrak{L}$ sur les réels apparaissant dans les calculs (par exemple une loi logarithmique) et par conséquent une loi conditionnelle aux intervalles d'arrondis; on peut par exemple prendre sur $I_{\tau(x)}$: τ"(x) = Z tel que $E[(Z-x)^2/\tau'(x)]$ soit minimum, où E désigne l'espérance mathématique.

En disposant de quelques échantillons de τ"(x) on peut alors estimer x avec une erreur plus faible qu'avec τ(x) et donner également une erreur quadratique moyenne sur cette nouvelle représentation. De cette représentation des réels découle une arithmétique stochastique définie à partir de la représentation des réels de manière identique aux arithmétiques précédentes.

Le problème est alors de savoir si cette arithmétique, intéressante pour la représentation des réels, est aussi performante pour la représentation des algorithmes. Cela n'est empiriquement justifiable que si la loi $\mathfrak{L}$ est stable par applications successives de l'arrondi τ" et d'un opérateur canonique. Cette propriété est approximativement vraie pour la loi logarithmique et les opérateurs de les plus usuels (addition, multiplication). Ainsi donc une arithmétique stochastique qui soit presque quantitativement "optimale" semble exister. C'est cette arithmétique que La Porte et Vignes [PV] proposent d'utiliser en 1974 dans le cas des arithmétiques flottantes, et de l'arrondi par excès et de la troncature.

Il reste à examiner les propriétés d'une telle représentation machine. Dans le cas de l'arrondi au plus près, cette arithmétique conduit à ne pas perturber et fait donc perdre les propriétés de l'arithmétique ensembliste.

Par ailleurs, le fondement d'une estimation précise de l'erreur avec de tels systèmes de représentation des nombres et peu d'échantillons est nous le verrons assez délicat [Fra]. L'intérêt quantitatif de tels systèmes de représentation machine n'est donc pas immédiat.

Par contre, il semble qu'une arithmétique stochastique basée sur une représentation aléatoire non triviale des nombres dans $\tau'(x)$ conserve les propriétés qualitatives de l'arithmétique ensembliste. Par ailleurs, contrairement à l'arithmétique d'intervalles cette arithmétique tient intrinsèquement compte des identités entre paramètres formels. Elle suit donc mieux la dynamique du calcul. De plus, l'utilisation d'un système assez large d'arithmétiques stochastiques permet, on le verra, d'effectuer un sondage qualitatif efficient des instabilités arithmétiques. Enfin, un tel système arithmétique est relativement simple à réaliser matériellement, et à un coût raisonnable. Cela constitue donc à notre avis un système arithmétique qualitatif qui devrait pouvoir être offert dans le futur aux utilisateurs afin de les aider à étudier leurs programmes de calcul. On signalera d'ailleurs qu' IBM [Buc] avait songé à la réalisation d'une telle arithmétique dès les années soixante.

C-3 Multiprécision.

Une autre manière de simuler les calculs plus exactement est d'utiliser des systèmes multiprécision. Le principe de ces systèmes est d'offrir la possibilité à l'utilisateur de représenter chaque nombre réel et chaque opérateur avec une précision paramétrée.

Cela permet par exemple d'exécuter les sections critiques d'un programme avec une précision plus forte pour limiter les erreurs d'arrondi [CH]. Il existe même des heuristiques pour régler automatiquement la précision de chaque opérateur afin d'obtenir une précision donnée sur un résultat d'un calcul [HV]. On peut ainsi calculer certaines constantes fondamentales avec une précision arbitraire.

Par ailleurs, la multiprécision permet d'étudier l'influence de la discrétisation machine sur un algorithme mathématique (convergence, stabilité) [Jon]. Ainsi ces systèmes arithmétiques offrent un grand intérêt pour le contrôle de l'erreur arithmétique.

Cependant l' inconvénient de tels systèmes est leur coût. Malgré le développement des technologies, il ne semble pas encore réaliste de les utiliser en temps réel.

Par contre, il semble que les outils multiprécision puissent être amenés à jouer un rôle important dans les phases de développement, que ce soit pour tabuler des constantes importantes ou dans le cadre d'un atelier d'audit de la qualité numérique.

C-4 Conclusion.

Les arithmétiques stochastiques et d'intervalles constituent donc des alternatives possibles aux arithmétiques usuelles pour représenter des calculs sur l'ensemble des réels. Ces systèmes proposés depuis longtemps dans la littérature s'avèrent de plus posséder des propriétés mathématiques intéressantes par rapport aux arithmétiques usuelles. Ainsi l'arithmétique d'intervalles permet d'obtenir des encadrements certains des résultats et, avec quelques raffinements, d'effectuer des preuves mathématiques non triviales à partir de résultats informatiques. Les arithmétiques stochastiques quant à elles se révèlent, nous le verrons, une aide précieuse pour étudier la stabilité arithmétique d'un calcul. Par ailleurs les systèmes multiprécision semblent pouvoir aider à augmenter la qualité des résultats informatiques.

L'évolution de la technologie rend maintenant ces systèmes réalistes [Mul]. Ils constituent donc peut être des bases futures d'un atelier de qualité du logiciel numérique.

D - CORRECTION A PRIORI DE L'ERREUR.

Malgré le développement d'autres systèmes arithmétiques la plupart des calculs resteront sans doute encore longtemps effectués en arithmétique flottante. A cet effet, des règles automatisables de transformation de programmes de calcul scientifique minimisant les instabilités arithmétiques en virgule flottante (la démarche devrait du reste être la même avec d'autres systèmes) doivent aussi être élaborées [Dav]. Là encore, il n'existe aucune théorie permettant d'aboutir à une transformation stabilisatrice de calculs.

Les numériciens utilisent de leur coté, un certain nombre de techniques empiriques pour rendre leurs programmes arithmétiquement plus stables. Les plus connues sont sans doute le préconditionnement pour les systèmes linéaires, l'application du schéma de Horner pour l'évaluation de polynômes, les resommations [Wol] (par exemple par ordre croissant des valeurs absolues) pour les calculs de sommes. Ces techniques éprouvées doivent évidemment être appliquées autant que faire se peut mais elles sont très spécifiques au problème traité et ne peuvent suffire pour aboutir à un outil de stabilisation automatique des programmes.

Pour arriver à un outil automatique, il faudrait être capable de reconnaître les formules instables arithmétiquement *a priori* dans les programmes de calcul et de les transformer pour empêcher l'erreur à l'évaluation.

Il faut pour cela arriver à définir un concept de forme réductible en accord avec l'idée qu'on se fait d'une formule pathologique arithmétiquement et trouver un algorithme de réductibilité stabilisant les formules basé sur des transformations simples. La réflexion des algébristes n'en est qu'à ses débuts. Les idées naïves échouent presque toutes. Ainsi, le compactage des calculs ne conduit pas forcément à les stabiliser.

Un des problèmes fondamentaux se posant est que la notion de formule arithmétiquement pathologique n'est pas attachée uniquement à une formule algébrique mais est aussi relative aux valeurs des variables. Par exemple, $\frac{1}{A} - \frac{1}{A+1}$ n'est pathologique que dans un voisinage de $A=+\infty$. On pourrait imaginer sa transformation en 1/A*(1/(A+1)), mais cette formulation ne serait valable qu'au voisinage de $+\infty$. Dans tout algorithme général de transformation, il faudra donc d'abord résoudre un système algébrique pour chercher les points où la formule à laquelle on applique l'algorithme est réductible (c'est à dire instable).

Ces problèmes mis à part, la littérature [Dav] [BV] fait cependant apparaître le phénomène de destruction comme le principal responsable de l'erreur d'arrondi. Si on ne s'intéresse qu'à ce phénomène, on est alors amené à définir l'instabilité arithmétique comme l'application du calcul approché à une formule du type : $F(X_1, \ldots, X_n)-G(Y_1, \ldots, Y_p)$ au voisinage de la "singularité" $F(X_1, \ldots, X_n) = G(Y_1, \ldots, Y_p)$.

Moyennant cette définition, on peut ébaucher quelques idées de stabilisation des programmes. Un algorithme de transformation pourrait se décomposer en deux étapes : Une phase de reconnaissance des instabilités "F(X1, ..., Xn) = G(Y1, ..., Yp)" suivie d'une phase de développement de F(X1, ..., Xn) - G(Y1, ..., Yp), par exemple par développement limité.

Le calcul symbolique apparait donc comme une voie possible de traitement de certaines instabilités. Les techniques à mettre en oeuvre pour un traitement symbolique apparaissent encore coûteuses et il reste à réfléchir sur la manière d'appréhender formellement les notions d'ordre de grandeur. Néanmoins, les travaux sur le traitement formel de l'instabilité arithmétique existent peu, et il serait sans doute intéressant de poursuivre dans cette voie.

CONCLUSION

Un examen critique des arithmétiques machines usuelles quant à leurs capacités de simuler des algorithmes continus de résolution de problèmes réels conduit donc à l'utilisation empirique d'arithmétiques discrètes stochastiques, d'intervalle ou multiprécision. Cette recommandation s'inscrit dans le cadre d'une tentative d'adaptation des ordinateurs et de la programmation pour résoudre des problèmes continus. Une démarche duale est aussi possible. Reveilles [Rev] part quant à lui d'une formulation discrète des problèmes sur les réels (analyse non standard) et propose alors une arithmétique machine taillée a cette formulation "universelle" des problèmes continus basée sur les calculs modulo un grand entier.

Une dernière question fondamentale est de réfléchir plus profondément sur la nature de ce qu'on cherche à représenter. On se place généralement dans l'optique d'une représentation globale de tous les réels en précision limitée. Vuillemin [Vuil] souligne à juste titre que seuls certains réels calculables nous intéressent. Il propose alors une méthode de calcul effectif en précision infinie sur l'ensemble des réels représentables par des algorithmes. Cette approche bien qu'encore peu réaliste d'un point de vue pratique est très convaincante d'un point de vue théorique.

BIBLIOGRAPHIE

[AC] AL et CODY, A proposed radix and worth length independent standard for floating-point arithmetic. IEEE Micro, vol 4, n°4, pp 86-100, 1981.

[BV] P BOIS et J VIGNES, Analysis of the numerical stability of algorithms. Rapport de l'Institut Français du Pétrole.

[Bin] P-M BINDER, Machine iteration of a linear function : local behaviour. Soumis à Comp&Maths with Appls. Octobre 1989.

[Bol] GERD BOLENDER, floating point computation with maximum accuracy, IEEE transactions on computers, vol C26, n°7, 1977.

[Bre] R P BRENT, On the precision attainable with various floating point number systems, IEEE trans on computers, vol C22, N°6, 1973.

[Bro] BROWN, A simple but realistic model of floating point computation, ACM transactions on mathematical software VOL 7, N° 4 pp 445-480, décembre 1981.

[Buc] W BUCHHOLZ, Planning a computer, Project Stretch, Mc Graw Hill, New york, 1962.

[Cla] DM CLAUDIO, contribution to the structure of computer arithmetic, Computing 24, pp 115-118, 1980.

[CH] M COHEN et T HULL, Toward an ideal computer arithmetic. IEEE proceedings of ARITH 8, 1987.

[Dav] JH DAVENPORT, Survey of symbolic applications for numerical computation. Copyright by the diamond consortium, 1987.

[Fra] P FRANCOIS, Mesure de l'erreur arithmétique et perturbations des calculs. Rapport de recherche IMAG-TIM3, Janvier 1989.

[Ger] G GERRITY, Computer representation of real numbers. IEEE transactions on computers, Vol C31, N°8, Aout 1982.

[GK] RT GREGORY et EV KRISHNAMURTY, Methods and application of error free computation. Springer Verlag, 1983.

[Gru] K GRUNER, Implementation of universal computer arithmetic with optimal accuracy. Computing 24, pp 181-193, 1980.

[HV] B HULSHOF et J VAN HULZEN, Automatic error cumulation control. Twente university of technology, Departement of computer science, The Netherlands.

[Jon] C JONES, A significance rule for multiprecision arithmetic. ACM transactions on mathematical software, vol 10, N°1, pp 97-107, 1984.

[Knu] KNUTH, The art of computer programming, vol 2. Addison Wesley.

[KM1] U W KULISCH et WL MIRANKER, The arithmetic of the digital computer, a new approach. SIAM rev, vol 28 n°1, mars 1986.

[KM2] U W KULISCH et WL MIRANKER, Computer arithmetic in theory and practice. Academic press, 1981.

[Mal] A MALCOM, On accurate floating point summation. IEEE communication of the ACM, vol 14, N°11, novembre1971.

[Mul] JM MULLER, Arithmétique des ordinateurs. Masson, 1989.

[Olv] F W OLVER, A closed computer arithmetic. IEEE proceedings of ARITH 8, 1987.

[Pic] M PICHAT, Contribution à l'étude des erreurs d'aarrondi en arithmétique virgule flottante. Thèse d'état, INPG-USMG, Grenoble, 1976.

[Por] M La PORTE, Une formulation numériquement stable donnant les racines réelles de l'équation du 3ième degrè, Rapport de recherche IFP N°21516, Octobre 1973.

[PV] M LA PORTE et J VIGNES, Méthode numérique de détection de la singularité d'une matrice. Numer. Math. 23, pp 73 -81. 1974.

[Rev] JP REVEILLES, Simulation arithmétique du continu, journées informatiques et mathématiques de Caen, actes, mars 1988.

[Rob] F ROBERT, Machine iteration for a linear function. Maths with Appls. 12 B, pp 1259-1274, 1986.

[SS] J. K SCHEIDT et C. W SCHELIN, Distribution of floating-point numbers. Computing 38, pp 315-324, 1987.

[Vig1] J VIGNES, New methods for evaluating the validity of the results of mathematical computations. Mathematiccs and computers in simulation, Transaction of IMACS, Vol XX, N°4, pp 227-249, Décembre 1978.

[Vuil] J Vuillemin, Communication aux journées informatiques et mathématiques de Caen, actes, mars 1988.

[Wol] J WOLFE, Reducing truncation errors by programming. Communication of the ACM, vol 7, N°6, juin 1964.

[Yoh] M YOHE, Roundings in floating point arithmetic. IEEE transactions on computers, vol C22, juin 1973.

CHAPITRE II

CONTROLE DES ERREURS D'ARRONDI

A - CONTROLE DE LA DERIVE NUMERIQUE.

Comme on l'a donc rappelé dans le premier chapitre, le résultat d'un calcul informatique en précision limitée et en particulier en arithmétique flottante peut, à cause des erreurs d'arrondi, ne pas du tout être représentatif du résultat du même calcul effectué en précision infinie.

L'emploi d'une arithmétique plus adéquate, plus précise, ou approchant mieux la structure du corps des réels et le respect de normes de programmation peuvent permettre de limiter certains problèmes. On pourra par exemple mettre en évidence certaines instabilités en utilisant une arithmétique stochastique, obtenir une grande précision sur des constantes critiques en employant des arithmétiques multiprécision ou mieux valider des phénomènes de convergence à l'aide d'une arithmétique d'intervalles. L'évaluation des polynômes selon le schéma de Horner ou des sommations dans l'ordre croissant des valeurs absolues seront aussi un facteur d'amélioration de certains résultats. Cependant, l'utilisation de tel ou tel système arithmétique ne permet pas de garantir systématiquement tous les résultats. Par ailleurs, en pratique, sauf pour quelques cas précis les calculs sont effectués en arithmétique virgule flottante.

Quant aux règles de programmation permettant d'améliorer l'évaluation des calculs, elles ne constituent qu'un facteur partiel de qualité des calculs et en aucun cas une garantie. De plus l'utilisation de compilateurs modifiant l'ordre de certaines exécutions en vue d'optimisations peut empêcher leur utilisation.

Ainsi, aussi bien parce qu'aucun système arithmétique et aucune règle de programmation ne permettent *a priori* de représenter tous les résultats que parce que les calculs sont effectués en pratique avec une arithmétique fixée, il importe de pouvoir mesurer les qualités arithmétiques des logiciels numériques, notamment en arithmétique flottante.

Les spécifications d'une méthode de qualification du logiciel peuvent être très variées, elles dépendent évidemment des degrés d'exigence et de réalisme de l'utilisateur.

Tout le monde voudrait disposer d'une méthode de correction des erreurs d'arrondi. Le moindre qu'on semble devoir exiger est de pouvoir décider qu'un logiciel est fiable. Mais on peut aussi chercher à repérer, mesurer ou expliquer les erreurs d'arrondi.

De nombreux auteurs ont travaillé sur la recherche de méthodes de qualification du logiciel numérique. On cherche ici à analyser les principaux travaux. La plupart des méthodes se proposent de déterminer une mesure de l'erreur commise entre un résultat informatique et le résultat exact, mais il est difficile d'isoler de grands principes théoriques. La majorité des méthodologies proposées dans la littérature comportent une bonne part d'empirisme. Seule une ligne directrice nous a paru s'imposer dans cette étude. Il s'agit de la séparation des méthodes en deux classes, les méthodes statistiques et les méthodes déterministes auxquelles on peut rajouter les heuristiques utilisées par tous les numériciens comme premiers tests de leurs programmes. Cette séparation articulera cette partie.

Nous insisterons particulièrement sur les approches statistiques en montrant quelques difficultés d'une approche paramétrique systématique notamment dès que la taille des problèmes est trop grande ou que certaines spécificités de l'arithmétique jouent, et en en proposant (tout au moins dans des cas particuliers) un certain nombre d'explications. Nous voulons éclairer dans ce chapitre la difficulté de fondement des méthodes de contrôle d'erreur autres que l'arithmétique d'intervalles.

Dans toute la suite, on reprendra le modèle d'algorithme mathématique et de représentation machine de celui ci présenté dans la première partie et on se placera dans le cas d'une arithmétique usuelle définie par un arrondi Δ, un ensemble de nombres machine S et des opérateurs canoniques $(f_j)_{j=1...n}$.

Un algorithme théorique sera donc représenté par un système dynamique discret sur les réels :

$$(A): \quad x_0 \text{ donné}, \quad x_{t+1} = f_t(x_t)\ t = 0...T(A) - 1$$

où $T(A) \in \mathbb{N} \cup \{\infty\}$, $x_i \in \mathbb{R}^{v(i)}$, et où les f_i sont des opérateurs canoniques de $\mathbb{R}^{v(i)}$ dans $\mathbb{R}^{v(i+1)}$.

L'image machine d'un tel algorithme est alors représentée par le système :

$$(A_m): \quad X_0 = \Delta(x_0), \quad X_{t+1} = F_t(X_t)\ t=0 \ldots T(A)-1, \quad \text{avec } F_t = \Delta f_t.$$

On se placera ici dans le cas où T(A) = T est fini.

Le problème du contrôle de l'erreur se présente alors mathématiquement de la manière suivante :
Contrôler $(X_t - x_t)$ (en particulier pour t = T).

B - CONTROLES DETERMINISTE ET EMPIRIQUE.

B-1 Evaluation empirique.

La plupart des utilisateurs du calcul scientifique n'utilisent pas de méthodes sophistiquées pour valider leurs résultats. Par contre, ils utilisent largement les possibilités offertes par leur matériel pour sonder empiriquement la qualité de leurs calculs. Le principe souvent utilisé est d'engendrer plusieurs résultats machine qui devraient *a priori* être représentatifs du résultat exact et de ne retenir le programme que si ils sont suffisamment proches.

Les résultats peuvent être produits soit en effectuant les calculs sur une même machine avec différentes précisions (usuellement la simple et la double précision) avec différents modes d'arrondi (au plus près, supérieur, inférieur) avec ou sans coprocesseur arithmétique ou encore en perturbant légèrement les données, soit en implantant l'algorithme sur différents ordinateurs.

Toutes ces heuristiques sont intéressantes à appliquer pour sonder empiriquement la robustesse d'un calcul avant d'employer des méthodes plus lourdes. La dernière technique a de plus un intérêt historique : le développement de l'utilisation des stations de travail pour la conception des logiciels a en effet, sans doute, beaucoup contribué à faire prendre conscience aux utilisateurs de la nécessité de valider les résultats produits par les gros calculateurs, souvent d'ailleurs parce que certaines stations calculent plus précisément grâce à l'emploi de la norme IEEE.

Cependant, ces tests empiriques ne peuvent fournir qu'une indication sur l'existence d'instabilités arithmétiques.

B-2 Evaluations déterministes de l'erreur.

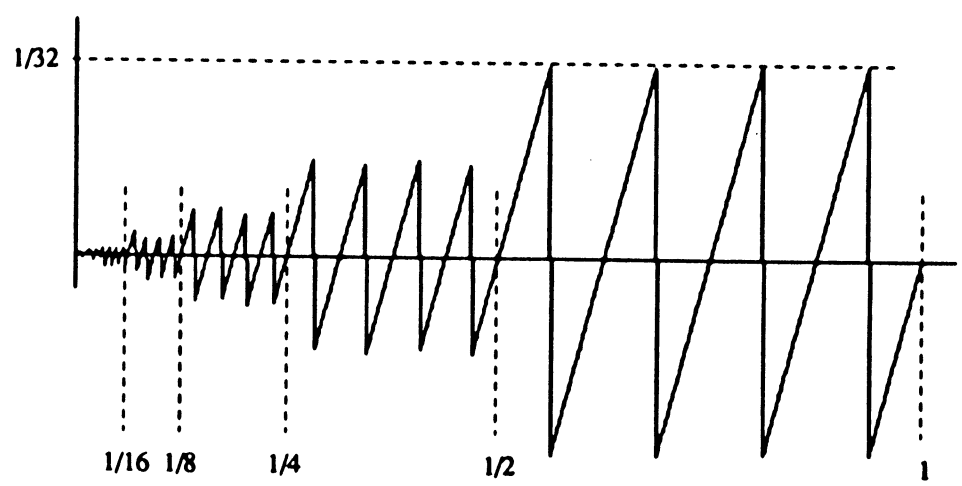
On a cherché dès les années cinquante à disposer de méthodes systématiques de contrôle d'erreur. L'approche qui s'est imposée la première et qui encore aujourd'hui est à la base de plusieurs produits utilisés pour le contrôle d'erreur (le logiciel ACRITH de Kulisch et Miranker commercialisé par IBM par exemple [Lan]) est une approche de type majoration déterministe de l'erreur. Il peut s'agir soit d'obtenir récursivement un majorant de l'erreur arithmétique finale soit d'encadrer systématiquement tout résultat informatique.

B-2-1 Méthodes de majoration.

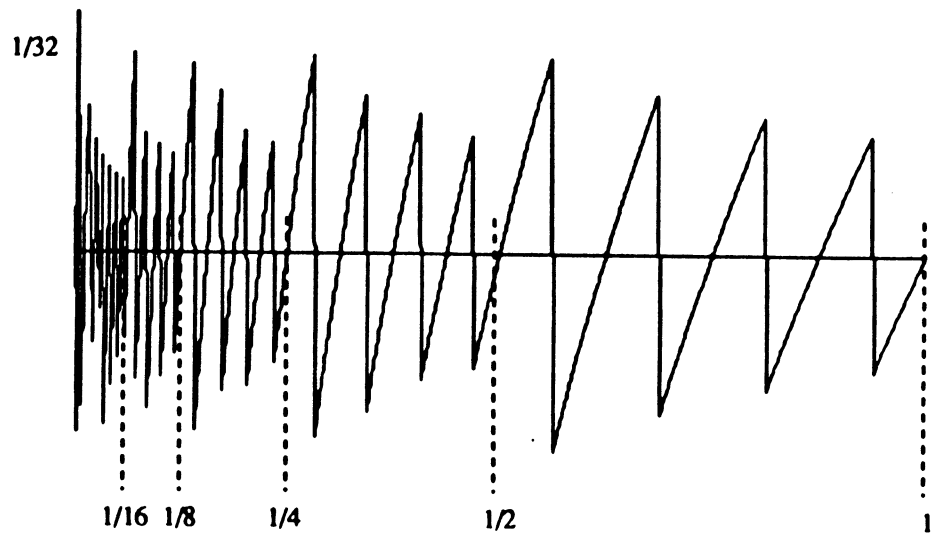
B-2-1-1 Evaluation de l'erreur d'arrondi élémentaire.

Le principe des méthodes de majoration récursive est de majorer l'erreur finale (entre le résultat machine et le résultat exact) à partir de majorations des erreurs élémentaires (entre les f_t et leurs images machine F_t) effectuées à chaque étape de calcul. Comme F_t est égal à Δf_t, l'erreur d'approximation de f_t par F_t est parfaitement déterminée par l'erreur d'arrondi sur $f_t(X_t)$. Le problème est donc de spécifier correctement l'erreur d'arrondi. Nous nous placerons en arithmétique à virgule flottante.

Dans ce cas, le problème vient du fait que cette erreur dépend du point d'arrondi de manière non triviale. Plus précisément, on ne peut ni la considérer comme une erreur relative, ni la considérer comme une erreur absolue. Alors que localement (entre deux nombres représentables de S) elle s'apparente à une erreur absolue, globalement elle ressemble à une erreur relative. Pour montrer ce phénomène, on visualise maintenant l'erreur commise en représentant un réel de [0,1] par son arrondi au plus près en base 2 avec des mantisses de 4 bits.



Erreur absolue d'arrondi en x = f(x)



Erreur relative d'arrondi en x = f(x)

On trouve diverses approches dans la littérature.

- Le cas de l'erreur relative est le plus simple. Pour cette spécification :

$$F_t(X_t) = f_t(X_t).\ (u_t + e_t)$$

où u_t désigne le vecteur de dimension v(t+1) dont toute les composantes sont égales à 1, "." désigne le produit tensoriel, et où e_t est un vecteur de taille v(t+1) dont toutes les composantes e_t^i vérifient :

$$-\beta^{-n} \leq e_t^i \leq \beta^{-n}$$

n étant un entier fixé dépendant du mode d'arrondi (par exemple la longueur de la mantisse en arrondi au plus près, la longueur de la mantisse moins un pour les arrondis vers zéro vers l'infini, par excès ou par défaut).

- Pour une spécification en erreur absolue, on obtient :

$$F_t(X_t) = f_t(X_t) + e_t$$

avec, pour tout i, $-\beta^{-m+\exp(X_t)} \leq e_t^i \leq \beta^{-m+\exp(X_t)}$, où m est égal à n+1, et où exp(x) désigne l'exposant de x dans sa représentation virgule flottante.

B-2-1-2 Majoration récursive.

Différents auteurs [Mil1], [Mil2], [LS1], [LS2] utilisent cette évaluation de l'erreur d'arrondi élémentaire pour obtenir des majorations de l'erreur arithmétique. Les méthodes automatisables sont basées sur une linéarisation du système dynamique "implanté " au voisinage de la trajectoire de référence (l'algorithme "idéal"), Miller propose ainsi, à partir d'un modèle d'algorithme numérique similaire au nôtre (cf notion de *computational graph*, [Mil]) et d'une formulation en erreur absolue, une méthode d'évaluation récursive de la propagation des erreurs d'arrondi qui débouche sur un logiciel d'étude de cette propagation. Larson et Sameh [LS1] adaptent cette méthode à une formulation en erreur relative et c'est celle que nous présentons ici.

On a $x_{t+1} = f_t(x_t)$ et $X_{t+1} = f_t(X_t).(u_t+e_t) = f_t(X_t).u_t+f_t(X_t).e_t = f_t(X_t)+f_t(X_t).e_t$, par conséquent :

$$\frac{X_{t+1}-x_{t+1}}{x_{t+1}} = \frac{f_t(X_t)}{x_{t+1}}.e_t + \frac{f_t(X_t)-f_t(x_t)}{x_{t+1}}$$

toutes les opérations sont prises ici au sens tensoriel. En négligeant les termes du second ordre relativement aux perturbations, il vient alors en notant $V_t = (X_t-x_t)/x_t$:

$$(M) \qquad V_{t+1} = e_t + E_t.V_t$$

où E_t est la matrice des élasticités de f_t prise au point x_t à savoir

$$E_t = \left[\frac{\partial \log f_t^i}{\partial \log x_t^j} \right] \quad (i,j) \in \{1,2, \ldots, v(t+1)\} \times \{1,2, \ldots v(t)\}$$

Cette classe de méthodes bien qu'automatisable ne retiendra pas notre attention. Outre les problèmes de justification théorique de celle ci (on ne retient que le premier ordre), elle nous semble impraticable, la complexité des calculs nécessaires pour évaluer l'erreur étant du même ordre que celle des calculs apparaissant dans l'algorithme. Les possibilités pratiques de l'outil de Miller seraient cependant à étudier précisément. On souligne également que cette méthode d'estimation récursive peut se révéler un outil pratique pour faire une évaluation sommaire sur "à la main" de la stabilité d'algorithmes simples (voir paragraphe III).

B-2-2 Méthodes d'encadrement.

Un deuxième type de méthodes utilisées pour le contrôle d'erreur arithmétique est basé sur l'utilisation d'une arithmétique d'intervalles [KM].

Comme nous l'avons vu au premier chapitre, le principe de l'arithmétique d'intervalles est de représenter chaque résultat par un intervalle machine le contenant.

Dans l'arithmétique d'intervalles "orthodoxe" chaque donnée (qui est un réel ou même un intervalle si on envisage le cas d'une erreur sur les données) est représentée par le plus petit intervalle machine (c'est à dire avec des bornes représentables en machine) la contenant et chaque opérateur canonique * appliqué à un pavé machine P est représenté en machine par le plus petit intervalle machine contenant *P.

Son utilisation permet donc d'obtenir un encadrement certain de n'importe quel calcul fini sans test ainsi que de son approximation machine (avec n'importe quel mode d'arrondi). Ainsi l'utilisation d'une arithmétique d'intervalles orthodoxe (De Moore) constitue une méthode sûre pour calculer une borne d'erreur sur la solution fournie par un calcul informatique.

Parallèlement, l'arithmétique d'intervalles est aussi utilisable dans de nombreux cas pour obtenir ou valider des résultats d'algorithmes itératifs. Les techniques de raffinement itératif et les algorithmes itératifs se transposent en effet souvent à l'algorithmique d'intervalles ([Rum], [KR], [KM], [Ada]) en donnant de plus des conditions de convergence simples (F(I) est inclus dans I, F étant la fonction itérée). L'arithmétique d'intervalles permet donc également l'évaluation de certaines erreurs numériques.

Elle semble donc devoir être un excellent outil de validation de résultats et pouvoir être un outil de base d'un atelier du logiciel numérique.

Cependant, même si les bornes d'erreur fournies par une arithmétique d'intervalles donnent un majorant fiable de l'erreur arithmétique, elles ne sont pas pour autant toujours satisfaisantes.

En effet, dans la pratique, les surestimations successives au cours de grands calculs conduisent parfois à un intervalle d'erreur du type $[-\infty, +\infty]$ et même si on sait avec certitude que le résultat y appartient c'est tout de même une maigre information en ce qui concerne l'erreur. Comme nous l'avons déjà souligné dans le chapitre précédent, la surestimation naturelle des résultats par l'arithmétique d'intervalles est de plus accentuée par le fait que cette arithmétique ne tient pas compte des identités formelles entre variables.

De nombreux travaux ont été effectués pour remédier à ces défauts.

Pour diminuer la surestimation de certains résultats, plusieurs auteurs proposent par exemple dans des cas particuliers des méthodes de validation basées sur des développements à l'ordre un des solutions [Ada]. Des réflexions ont également été menées dans le cadre du projet DIAMOND [Dav] pour éliminer le problème des identités formelles. Elles conduisent entre autres à transformer systématiquement toute formule polynomiale sous une forme de Horner. Pour l'exemple $(x-x^2)([0,1])$ du premier chapitre, cela conduit à l'intervalle [-1,0] au lieu de [-1, +1].

Ces travaux n'apportent pas de solution miracle à la surestimation de l'erreur, mais ils ont permis d'obtenir de nombreux succès [Ada]. Différents environnements de contrôle d'erreur basés sur l'arithmétique d'intervalles et intégrant plus ou moins ces raffinements existent maintenant [BNR]. Hormis les problèmes d'implantation posés par ces produits soulevés par plusieurs auteurs [KL], il nous semble intéressant de pouvoir utiliser un tel outil d'arithmétique d'intervalles maintenant entré en phase industrielle.

B-3 Conclusion.

Ainsi l'utilisation d'un logiciel d'arithmétique d'intervalles permet d'obtenir une majoration certaine de l'erreur arithmétique entâchant un calcul. Cependant, cette information est parfois peu précise. Il serait donc intéressant de disposer de méthodes de contrôle d'erreur plus fines.

C – APPROCHES STATISTIQUES.

Parallèlement à l'approche déterministe, on trouve dans la littérature de nombreuses méthodes de contrôle de l'erreur arithmétique basées sur une modélisation statistique de l'arrondi. Ces méthodes peuvent se répartir en trois classes :

- Celles basées sur des modèles probabilistes globaux de l'erreur d'arrondi, qui cherchent à rendre compte de l'aléa des données, et dont l'utilité réside plutôt dans l'évaluation des performances d'un algorithme ou d'un système arithmétique.
- Celles fondées sur une modélisation probabiliste d'un calcul machine ([Hen2], [HS], [PV1]) qui cherchent à représenter statistiquement la propagation des erreurs dans un calcul.
- Enfin les méthodes utilisant des perturbations aléatoires des calculs dont le but est de contrôler statistiquement la robustesse des résultats par rapport aux arrondis ([PV2],[VIG1], [Bru], [Fra]).

On examine ici ces méthodes en discutant leurs fondements et leur contribution à la compréhension des phénomènes.

C-1 Modèles globaux de l'erreur arithmétique.

Le but de départ de ce type de modèles est de chercher à mesurer les erreurs d'arrondi apparaissant dans un système informatique, plus finement qu'avec la mesure usuelle du pire des cas [Bre]. La surestimation fréquente de l'erreur par le pire des cas s'explique par le fait que cet estimateur ne prend pas en compte la fréquence d'apparition des nombres. En particulier, il ne peut alors pas représenter les phénomènes de compensation d'erreur qui arrivent en pratique.

On peut faire une analogie avec le calcul de la complexité d'un algorithme. Par exemple, une mesure de la complexité de l'algorithme d'Euclide de calcul du PGCD de deux nombres de taille N par majoration donne N opérations alors qu'en moyenne cet algorithme utilise LogN opérations [Knu].

L'idée est donc de mesurer la performance d'un système du point de vue de l'arithmétique en prenant en compte la fréquence d'apparition des nombres.

C-1-1 Modèles probabilistes de la distribution des nombres utilisés.

C-1-1-1 Distribution naturelle des nombres manipulés

Plusieurs auteurs ont essayé d'obtenir ou de justifier une distribution des nombres apparaissant dans les calculs [Knu], [SS], [Ham]. De quelques hypothèses, ils infèrent une distribution P pour la variable aléatoire X des nombres apparaissant dans des calculs.

- Knuth [Knu] suppose comme hypothèse :

  (A1) : P(cX) = P(X), pour tout réel c (invariance par changement d'échelle).

• On pense qu'il faudrait exiger plus naturellement que la distribution soit invariante par les opérations de base, c'est à dire que pour tous X et Y ayant pour distribution P on ait :

(A2) : P(X & Y) = P(X) &P(Y), pour tout opérateur canonique machine & .

Nous nous restreindrons à & élément de {+, *}. La plupart du temps, les opérateurs traités (opérations et lois discrètes) sont approchés par des opérateurs continus.

• De (A1) Knuth [Knu] déduit une distribution logarithmique des mantisses dont on rappelle ici la définition déjà donnée dans le premier chapitre. On dira que les mantisses m sont logarithmiquement distribuées en base b si :

$$p(m \leq x) = \frac{\log x + \log b}{\log b}$$

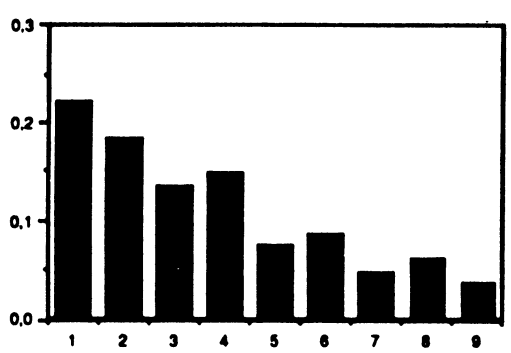
• Scheidt et Schelin [SS] montrent qu'une distribution logarithmique sur l'intervalle [m,M] des nombres représentables en machine, c'est à dire une fonction de répartition :

$$F_X(x) = \frac{\log x - \log m}{\log M - \log m}$$

implique une distribution logarithmique des mantisses, et que cette distribution est stable par produit et presque inchangée par addition, donc qu'elle vérifie approximativement l'hypothèse (A2). On peut également prouver que la seule distribution inchangée par produit est la distribution logarithmique. La distribution logarithmique est donc la seule loi susceptible d'être prise en compte pour nos mesures. La plupart des travaux concluent à celle ci.

C-1-1-2 Distribution des nombres produits lors d'un calcul

Expérimentalement, si on prend des nombres aléatoires et si on leur applique suffisamment d'opérations, on observe également une mantisse du résultat approximativement logarithmiquement distribuée. L'astronome Schlessinger [Sch] avait déjà remarqué cette propriété en observant des tables de calculs au début du siècle. En 1982, Turner [Tur] a prouvé la convergence rapide vers une distribution logarithmique par application de multiplications. Cette convergence peut du reste, facilement être mise en évidence en lisant une simple table de multiplication en base 10 pour laquelle l'histogramme du premier chiffre du résultat est donné ci dessous :



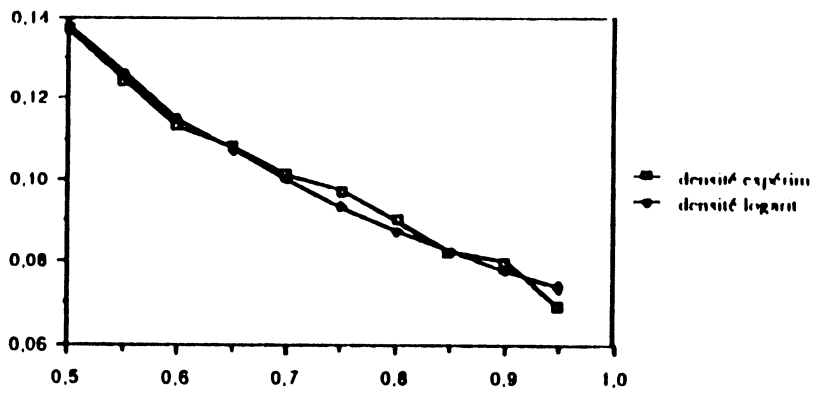
Fréquence d'apparition des différents chiffres comme chiffre de tête dans la table de multiplication

Pour l'addition ou a fortiori des calculs quelconques, la convergence ne peut être prouvée mais elle constitue une réalité expérimentale. Nous le visualisons par deux expériences :

•La première expérience consiste à remplir un tableau T de 100000 réels tirés aléatoirement suivant une loi uniforme, puis à effectuer l'algorithme

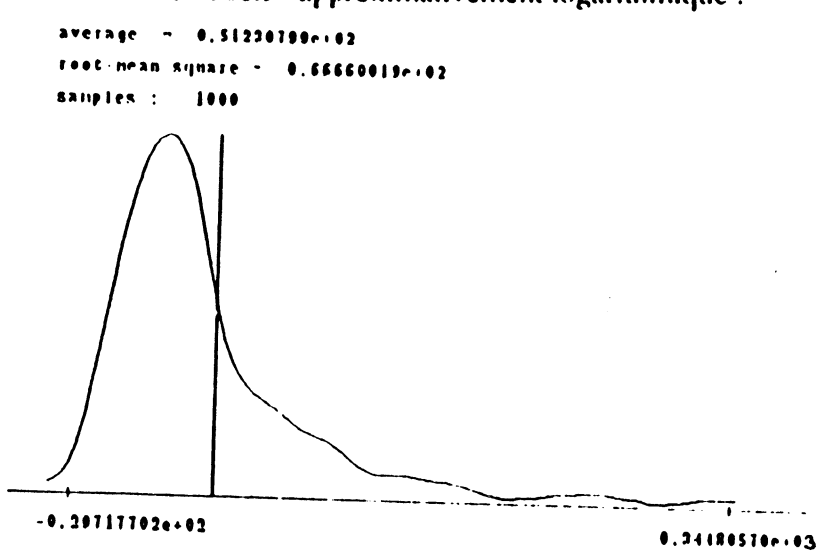
Pour i=1 jusqu'à 5 faire
Pour j=1 jusqu'à 100000 faire
T[j]:=T[random] + T[random]

Où random est une fonction donnant un entier aléatoire entre 1 et 100000.
Le graphe ci dessous présente la distribution des mantisses obtenue comparativement à la distribution logarithmique :



Distribution quasi logarithmique de la mantisse de la somme de 5 variables uniformes .

•La deuxième expérience consiste elle à étudier une variable aléatoire du type (...((x_1 $\&_1$ x_2) $\&_2$ x_3) .....) $\&_{n-1}$ x_n où les x_i suivent une loi uniforme sur [1,2] et $\&_i$ = +,-,*,/ avec une probabilité 1/4. Nous constatons pour n=100 et 1000 échantillons que la répartition des nombres ainsi générés suit une loi "à l'oeil" approximativement logarithmique :



Ainsi donc, la distribution logarithmique des mantisses peut également être prise comme modèle pour l'évaluation d'opérateurs sur des données perturbées.

C-1-1-3 Simplifications

La distribution logarithmique bien qu'intéressante d'un point de vue théorique est assez complexe. De plus, en pratique, on s'intéresse souvent seulement à l'erreur d'arrondi *locale*, c'est à dire à [X - r(X)], où r(X) est la représentation machine de X. Il est donc intéressant de pouvoir disposer d'une distribution de ces erreurs.

Feldstein et Goodmann [FG] ont prouvé que conditionnellement à l'intervalle machine, les erreurs d'arrondi convergent (en arithmétique "virgule flottante", sous l'hypothèse d'une distribution logarithmique des mantisses) vers une loi uniforme quand la taille de la mantisse tend vers l'infini. Cette convergence est de plus assez rapide (5 ou 6 bits suffisent en base 2). Beaucoup de travaux adoptent donc ce modèle approximatif pour les erreurs locales. Nous discuterons de cette utilisation ultérieurement.

C-1-2 Estimation de performances

C-1-2-1 Evaluation des systèmes arithmétiques

Sous ces hypothèses probabilistes globales [FG], [Knu], différents auteurs ont comparé les performances respectives des systèmes de représentation flottante et logarithmique des nombres à taille du mot mémoire fixée [Cod],[Bre], et pour différents modes d'arrondi [Cod]. Le système logarithmique est le meilleur des systèmes étudiés et parmi les systèmes virgule flottante, la base 2 avec convention du premier "1" implicite est optimale. L'apport des modèles probabilistes dans ce domaine est très claire. L'utilisation d'un modèle probabiliste permet par exemple, au contraire des modèles déterministes, de discriminer entre la base 2 et la base 4 [Cod].

C-1-2 -2 Evaluation d'opérateurs

En considérant des données aléatoires et en supposant une distribution des erreurs d'arrondi intermédiaires, différents auteurs ont également estimé les performances d'algorithmes classiques du point de vue des erreurs d'arrondi en arithmétique à virgule flottante [BF], [KL2], [RS]. Ces auteurs partent presque tous d'une distribution pour l'erreur relative d'arrondi qu'ils précisent en faisant [BR] ou sans [RS] faire le lien avec les modèles globaux. Ils supposent de plus les différentes erreurs d'arrondi *non corrélées* et *les arrondis locaux sans biais*.

Exemple.

On a par exemple étudié une telle modélisation pour la somme et le produit en arithmétique flottante avec arrondi au plus près. On rappelle que exp(X) représente l'exposant de X.

On a considéré trois cas :

1) La somme S de n nombres aléatoires uniformément distribués sur [a, b].

2) La somme S de n-1 nombres aléatoires X_i i = 2,...,n-1 uniformément distribués et d'un nombre X_1 tel que $\exp(X_1+X_2+ \ldots +X_n) = \exp(X_1)$.

3) Le produit P de n nombres aléatoires uniformément distribués sur [a, b].

On propose les modèles probabilistes qui suivent :

Modèles proposés.

• Pour les exemples 1et 3, la $i^{ème}$ étape de calcul est modélisée par :

$$R_{i+1} = (R_i \,\&\, X_{i+1}) * (1 + e_{i+1}),\ i = 1...N-1.$$

avec $E(e_{i+1}) = 0$, $E(e_{i+1}^2) = \delta^2$, et e_i non corrélée avec R_i et e_j pour $j \neq i$. & est l'addition dans le cas du premier exemple, et la multiplication dans le cas du troisième.

• Pour le deuxième exemple, on modélise la ième étape du calcul par :

$$R_{i+1} = R_i + X_{i+1} + e_{i+1},\ i = 1...N-1$$

avec $E(e_{i+1}^2) = \delta^2$ et e_i non corrélée avec R_i et e_j pour $j \neq i$.

On estime l'erreur arithmétique E par $\sqrt{E[(R_n - S)^2]}$ pour les sommes et par $\sqrt{E[(\frac{R_n - P}{P})^2]}$ pour le produit. Par la suite, σ et μ dénoteront respectivement l'écart type et la moyenne des variables X_i.

Théorème.

On donne ci dessous le terme principal en δ pour les erreurs arithmétiques E_i correspondant aux trois modèles :

$$E_1 = \sqrt{\frac{2n^3\mu^2 + (3\sigma^2 - 15\mu^2)n^2 + (9\sigma^2 - 17\mu^2)n + (30\mu^2 - 12\sigma^2)}{6}} * \delta$$

$$E_2 = \sqrt{(n-1) * \beta^{\exp(X_1)}}\ \delta'$$

$$E_3 = \sqrt{(n-1)} * \delta$$

Démonstration : Il s'agit de simples récurrences.

Remarques :

• Si μ est différent de 0, le terme principal en n de E_1 est $E_1 \cong \frac{n^{\frac{3}{2}}}{\sqrt{3}} * \mu * \delta$. On retrouve la valeur donnée par Robertazzi et Schwartz [RS].

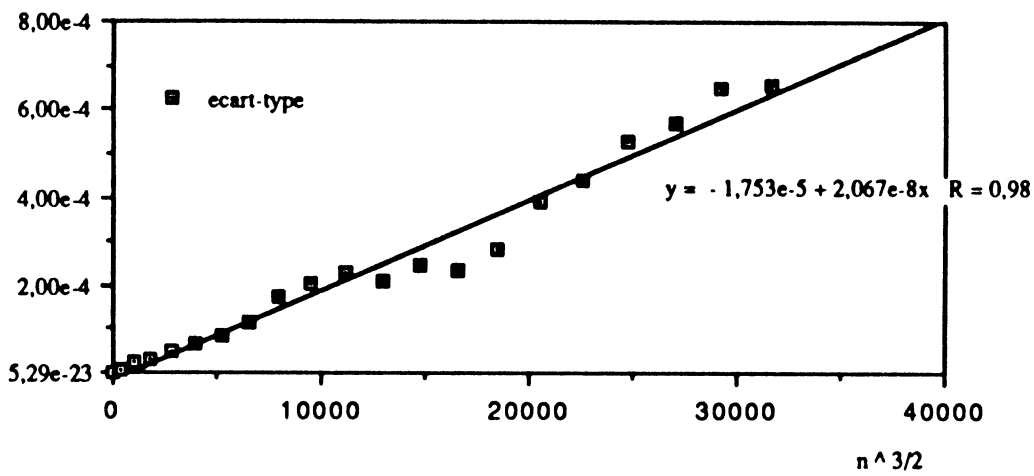
• En pratique, pour l'arithmétique flottante en arrondi au plus près, on prend (en se référant aux modèles probabilistes, voir les travaux de Brent [Bre]) :

$$\delta = \frac{\sqrt{(\beta^2 - 1)}}{\beta^L * \sqrt{24 * \log(\beta)}}$$

où β est la base du système de représentation et L la longueur des mantisses[BR]. Dans le cas de la simple précision IEEE (Base 2, L = 24) cette formule donne $\delta = 8.59\ 10^{-8}$.

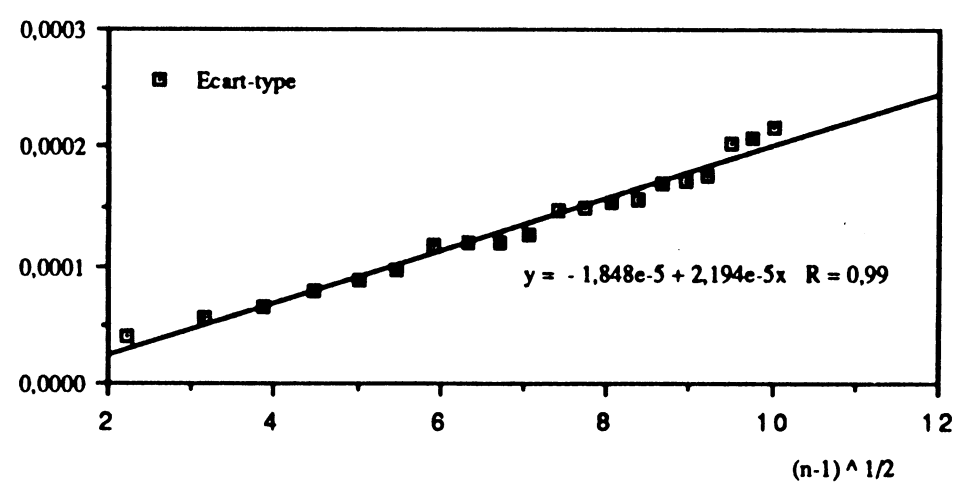
• Pour δ' (toujours en arrondi au plus près) on prendra $\delta' = \frac{\beta^{\exp(X_1)-1-L}}{\sqrt{3}}$, ce qui donne pour la simple précision et pour $X_1 = 1000$ une valeur de $1{,}76\ 10^{-5}$.

• On visualise par exemple la bonne adéquation de l'estimateur probabiliste d'erreur en $a.n^{3/2}$ du premier modèle, pour de petites tailles, dans le cas de la somme de variables aléatoires sur [1,2] :



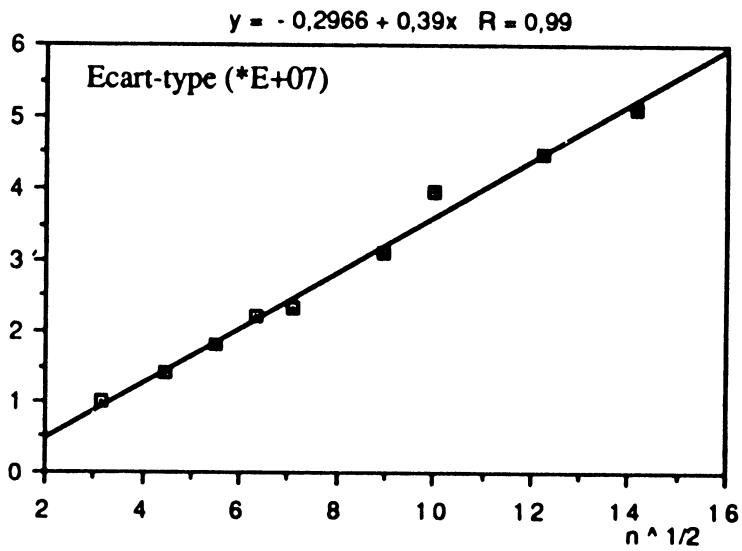
Adéquation du modèle d'erreur absolue sur la somme de nombres aléatoires de même ordre de grandeur.

• Le graphe ci dessous montre que pour $X_1 = 1000$ et variables uniformes sur [1,2] le deuxième modèle en $a(n-1)^{1/2}$ est de même satisfaisant (sur l'écart type) pour des tailles inférieures à 100 :



Adéquation du modèle d'erreur sur la somme de n-1 nombres de même ordre de grandeur et d'un nombre grand devant eux.

• On visualise également ci dessous la bonne adéquation du modèle de produit pour des variables uniformes sur [1,2]



Adéquation du modèle en $n^{1/2}$ pour l'erreur relative sur le produit de n nombres de même ordre de grandeur.

Propriété.

Les termes principaux E_i^{∞} en epsilon machine $\varepsilon = 2^{-L}$ des estimations usuelles des erreurs L^{∞} correspondant aux trois cas étudiés sont donnés par les formules suivantes :

$$E_1^{\infty} = \frac{(n^2+n-2)}{4} * e_{\infty} * \varepsilon$$ où e_{∞} est un majorant des valeurs absolues des X_i.

$$E_2^{\infty} = (n-1) * \exp(X_1) * \frac{\varepsilon}{2}$$

$$E_3^{\infty} = (n-1) * \frac{\varepsilon}{2}$$

Ainsi si l'adéquation des estimateurs statistiques proposés dans le théorème précédent persiste pour de larges tailles des calculs, leur utilité augmente avec la taille des calculs.

C-1-3 Problèmes de tels modèles globaux.

C-1-3-1 Critique pratique.

Cependant, si cette approche est assez convaincante pour l'évaluation des qualités de systèmes de représentation des nombres, elle parait moins intéressante et surtout moins utilisable pour valider systématiquement des algorithmes. Tout comme dans le cas des méthodes de majoration récursive, même sous les hypothèses faites, l'expression de l'erreur globale n'est en effet, dans le cas général, pas facile à manipuler. Par ailleurs, pour l'utiliser, on doit effectuer une linéarisation [RS], [KL2], et cette linéarisation n'est pas justifiée *a priori*. En fait cette approche n'a été utilisée que pour des algorithmes très linéaires et relativement simples comme des algorithmes de calcul de sommes [RS] ou de transformées de Fourier [BF], [KL2].

C-1-3-2 Critiques théoriques.

De plus, même lorsqu'un tel modèle d'évaluation d'erreur est utilisable son application ne nous semble pas si facile à justifier.

Il convient d'abord de souligner que le modèle logarithmique qui sert de base sous-jacente à une telle modélisation probabiliste n'est qu'approximatif. Pour justifier ce modèle, la distribution discrète des nombres est par exemple supposée continue. Knuth [Knu] montre qu'en toute rigueur cette approximation n'a pas de sens mathématique. En fait la distribution logarithmique des mantisses, puisqu'approximative, n'est valable que sur les premiers chiffres. Il y a donc quelque paradoxe à vouloir en déduire une information sur les derniers bits. Cela rend en particulier l'utilisation pratique du résultat de Feldstein et Goodman [FG] (qui déduisent une distribution uniforme sur les derniers bits d'une distribution logarithmique des mantisses) contestable à la base bien que mathématiquement correct. Les différentes approximations successives utilisées conduisent d'ailleurs à des contradictions avec les axiomes originaux; puisque si la distribution logarithmique est pratiquement inchangée par addition, ce n'est pas le cas de la distribution uniforme. Cependant, le fait qu'une telle hypothèse soit non justifiée serait sans conséquence si les estimateurs statistiques étaient suffisamment stables.
Or il se trouve que les performances des estimateurs statistiques proposés dans la littérature ne résistent pas toujours à une approximation des hypothèses statistiques et en particulier à l'introduction d'un biais, même léger. Pour les modèles statistiques de sommes et produits donnés antérieurement, supposer un biais sur les erreurs d'arrondi locales inverse par exemple les performances relatives des estimateurs statistiques et de l'estimateur du pire des cas pour un nombre de calculs suffisamment grand. Le théorème suivant étaye cet argument :

Hypothèse.

On modifie légèrement les modèles du paragraphe précédent. On suppose maintenant $E(e_i) = m \neq 0$. Ce modèle pourrait être plus réaliste dans certains cas [MM]. On donne sous cette hypothèse des bornes inférieures sur les erreurs quadratiques E_i.

Théorème.

Les erreurs quadratiques E_i au premier ordre en ε vérifient :

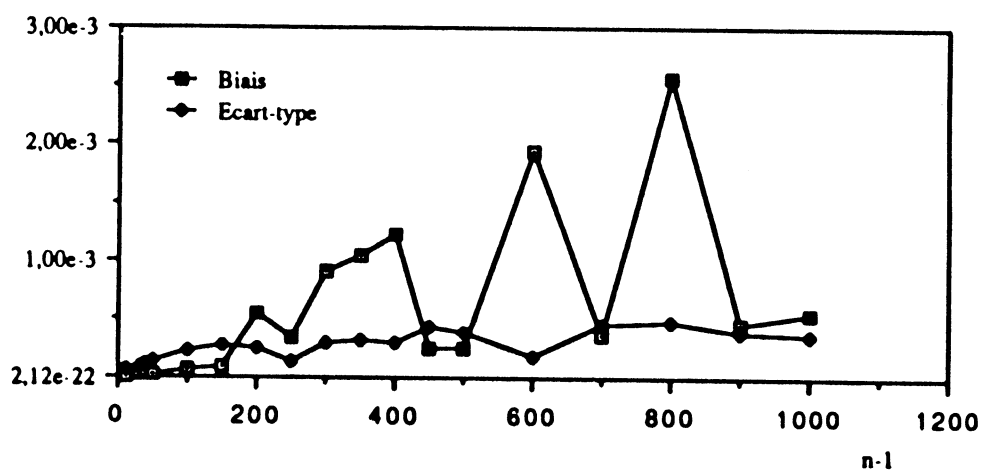
$$E_1 \geq \frac{n^2+n-2}{2} * |m*\mu|$$

$$E_2 \geq (n-1) * |m*e(X_1)|$$

$$E_3 \geq (n-1) * |m|$$

Une légère modification du modèle est donc suffisante pour invalider les estimateurs statistiques du paragraphe précédent pour de grandes tailles.

On visualise d'ailleurs dans le deuxième exemple le bien-fondé d'un tel argument de biais en montrant l'inadéquation d'un estimateur basé sur l'absence de biais dès que la taille du problème est suffisante :



Importance du biais dans l'erreur arithmétique sur une grande somme

C-1-4 Conclusion.

L'approche probabiliste globale rompt donc avec l'approche traditionnelle du pire des cas, purement déterministe, et introduit une mesure *a priori* plus intéressante et plus informative de l'erreur (la mesure est reliée à la probabilité d'avoir une erreur donnée). Par ailleurs, elle permet d'obtenir des résultats qualitatifs intéressants dans des cas pratiques importants (transformées de Fourier, sommes). Mais dans aucun cas (à cause des différentes approximations qui sont faites), nous ne pensons qu'elle peut être utilisée pour tous les types de calculs et pour toutes tailles de problèmes.

C-2 Modèles probabilistes de l'erreur dans un calcul.

Au delà de l'évaluation des qualités d'un système de calcul, il est important de comprendre l'influence des erreurs d'arrondi sur le résultat d'un calcul donné. Les approches probabilistes apparurent pour essayer de remédier à deux gros défauts des méthodes déterministes : leur tendance à majorer l'erreur et leur limitation informative. Différents auteurs (Henrici [Hen1], Hull et Swenson [HS], La Porte et Vignes [PV1]) ont proposé des modèles probabilistes de l'arrondi lors d'un calcul. Ces modèles consistent pour l'essentiel à représenter les erreurs d'arrondi locales par des variables aléatoires. Cette approche semble avoir deux avantages. Premièrement, celui de conserver la diversité du phénomène. Deuxièmement, celui de visualiser un comportement moyen.

C-2 -1 Modèle d'Henrici.

Henrici [Hen1][Hen2] s'est intéressé à la discrétisation des équations différentielles, en arithmétique fixe ou éventuellement flottante. Considérons par exemple l'équation :

$$y' = f(y, t)$$

Et l'algorithme destiné à le résoudre par le schéma de discrétisation d'Euler Cauchy :

$$y_{n+1} - y_n = h * f(y_n, t_n)$$

Le calcul de ce schéma sur ordinateur est entaché d'erreurs d'arrondi, on exécute en fait :

$$Y_{n+1} - Y_n = h * f(Y_n, t_n) + e_n$$

où e_n est l'erreur d'arrondi à l'étape n.

Si on linéarise la trajectoire obtenue au voisinage de la véritable trajectoire, on obtient alors une équation différentielle discrétisée de l'erreur arithmétique entre la solution calculée et la solution exacte.

$$E_{n+1} - E_n = h * g(E_n, t_n) + e_n$$

Henrici [Hen1] suppose que e_n est un "bruit blanc " (c'est à dire que les différentes erreurs d'arrondi sont non corrélées), doté d'une certaine stationnarité (e_n est uniforme en arithmétique fixe). Asymptotiquement (sur le pas de discrétisation) et en négligeant les erreurs de méthode, il obtient alors une équation différentielle stochastique qui donne par résolution, un modèle de propagation d'erreur. Le modèle d'Henrici est très intéressant d'un point de vue théorique; d'abord parce qu'il permet de représenter qualitativement la propagation des erreurs lors de l'intégration d'un système différentiel (en liant la trajectoire des erreurs à la trajectoire intégrée) ensuite parce qu'il est suffisamment paramétrique pour s'ajuster aux résultats pratiques. Malgré cet intérêt, les travaux sur ce modèle n'ont pas été poursuivis. Cela peut se comprendre par le fait que cette séduisante théorie de la propagation des erreurs d'arrondi est beaucoup trop complexe pour être appliquée à un gros problème réel.

C-2-2 Modèle de Hull et Swenson.

Hull et Swenson [HS] proposent eux un modèle empirique général de propagation des erreurs. Le principe de celui-ci est de voir les erreurs d'arrondi élémentaires comme des variables aléatoires uniformes conditionnellement à chaque intervalle d'arrondi. On représente ainsi le calcul machine par une perturbation aléatoire (à chaque étape uniforme et sans biais) du calcul exact. L'erreur est alors supposée être bien représentée par la variance σ de cette variable aléatoire.

Pour tester leur modèle (du point de vue de l'erreur) en simple précision, Hull et Swenson le simulent en double précision. Dans leurs expériences, l'erreur entre le résultat machine et la variable aléatoire modèle est en $o(\sigma)$ ainsi que la différence entre l'erreur d'arrondi et l'écart-type de la variable aléatoire modèle, et la différence entre la moyenne de la variable aléatoire modèle et le résultat double précision. Il y a donc du point de vue de l'estimation de l'erreur une bonne adéquation à la réalité.

Hull et Swenson proposent alors une interprétation de leurs résultats. Pour eux, la variable modèle est Gaussienne, centrée sur le vrai résultat. Le résultat du calcul est ainsi modélisé par le tirage d'une variable aléatoire normale centrée sur le vrai résultat. Cela permet de proposer un intervalle de confiance pour le calcul basé sur la variance de la variable aléatoire modèle. Ce travail de Hull et Swenson est d'un grand intérêt historique, notamment parce qu'il souligne les difficultés d'une modélisation probabiliste de l'erreur d'arrondi en virgule flottante (liées aux ordres de grandeur des nombres et à la spécificité des opérateurs machine) et parce que le modèle normal a constitué un archétype de modélisation de l'arrondi. Par contre, ces travaux empiriques de Hull Swenson ne débouchent pas sur une véritable méthode de contrôle d'erreur.

C-2-3 Modèle de La Porte et Vignes.

La Porte et Vignes [PV1] proposent également un modèle probabiliste pour le contrôle de la solution d'une équation algébrique ou d'un système linéaire, cas particuliers pour lesquels le problème à résoudre admet un résidu, c'est à dire peut s'écrire sous la forme $f(x) = 0$.

Les deux auteurs représentent d'abord le résultat machine X par $x(1+a)$ où a est uniforme. et où x est une approximation de X à la précision machine près. Ils modélisent ensuite de manière probabiliste le calcul du résidu f(X) en machine, en supposant les erreurs relatives d'arrondi effectuées à chaque étape uniformément distribuées. Ils en déduisent alors un test statistique de la solution à la précision machine ($x = x_0$ tel que $f(x_0) = 0$)) basé sur le seul résidu machine.

Ce modèle à l'avantage de fournir des tests pratiques de validation de résultats. Cependant, ce type d'approche n'est envisageable que dans des cas ou la fonction f est relativement simple (dans le travail de La Porte et Vignes les fonctions de validation s'expriment comme des sommes de produits).

C-2-4 Conclusion.

Même s'ils constituent un point de vue théorique intéressant, les modèles probabilistes classiques d'un calcul présentés ici sont, pratiquement, peu convaincants. D'une part parce qu'ils ne débouchent pas sur des méthodes pratiques de contrôle des erreurs d'arrondi. D'autre part parce qu'ils sont purement empiriques et de test difficile.

C-3 Méthodes de perturbation.

Une dernière classe de méthodes [CB], [Cha3], [PV2], [Vig1], [Vig2] de contrôle de l'erreur arithmétique en arithmétique virgule flottante est basée sur l'utilisation de perturbations aléatoires des calculs. L'idée d'utiliser des perturbations pour contrôler la stabilité est ancienne [For], [Buc] mais elle est longtemps restée purement empirique et cherche encore en partie ses fondements.

Au début, les perturbations étaient employées sans aucune interprétation pour tester empiriquement la stabilité des programmes. En 1974, La Porte et Vignes utilisent cette idée pour corriger l'erreur en troncature ou en arrondi par excès [PV2] et proposent une arithmétique stochastique corrective facile à implanter. Ils adaptent ensuite [PV2] leur perturbation à l'arrondi au plus près et mettent au point une méthode "CESTAC " d'évaluation de l'erreur d'arrondi (associée à leur arithmétique) basée sur une interprétation empirique des perturbations [FV].

En 1988, Chesnaux [Che1] donne un modèle de la perturbation de La Porte et Vignes pour les algorithmes finis justifiant l'utilisation de CESTAC sous certaines hypothèses. F. Chatelin et M.C. Brunet proposent dans le même temps le modèle PRECIX [CB] pour les algorithmes QR et de Gauss, et utilisent le même type de perturbations.

C-3-1 Arithmétiques stochatiques.

Perturber les calculs consiste essentiellement à remplacer chaque opérateur machine & par une perturbation aléatoire de celui ci $\&_p$. Avec une telle arithmétique stochastique, un résultat de calcul est une variable aléatoire X.

Dans la pratique, cela est réalisé en perturbant chaque résultat produit par la machine. L'opérateur perturbé $\&_p$ associé à un opérateur réel & sur $\mathbb{R}^v$ est ainsi défini par :

$$\&_p(x) = p(\&x) \text{ où p est une perturbation des réels.}$$

La plupart des auteurs utilisent une perturbation au dernier bit :

$$p_{t,q}(x) = x + \alpha(t, q).\ 2^{\exp(x)-L}$$

où α est une variable aléatoire sur $\{-1, 0, 1\}$, telle que prob $(\alpha = -1) = q$, prob $(\alpha = 1) = t$, et prob $(\alpha = 0) = 1 - t - q$.

Vignes [PV1] prend $t = q = 1/4$ pour l'arrondi au plus près, $t = 0$ et $q = 1$ pour l'arrondi supérieur, $t = 1$ et $q = 0$ pour la troncature. Dans les deux derniers cas l'évocation des modèles probabilistes globaux l'autorise à présenter sa perturbation comme une méthode de correction d'arrondi (en moyenne sur tous les calculs). Chatelin [Cha3] prend $t = q = 1/4$ et utilise l'arrondi au plus près.

C-3-2 Contrôle d'erreur paramétrique.

La plupart des utilisations d'arithmétiques stochastiques sont basées sur l'inférence de lois de probabilité pour X par interprétation de l'effet des perturbations. De telles propriétés permettent ensuite de justifier des estimateurs statistiques de l'erreur arithmétique.

C-3-2-1 CESTAC.

L'estimation de l'erreur arithmétique par la méthode CESTAC (Contrôle et Estimation Stochastique des Arrondis de Calcul) de Vignes est basée sur l'hypothèse d'une distribution approximativement normale centrée sur le vrai résultat x de X (voir [FV]).

C-3-2-1-1 Principe de la méthode de Vignes.

Sous cette hypothèse, Vignes [FV] propose d'estimer x avec 3 échantillons de X par la moyenne empirique des trois échantillons et d'évaluer le nombre de chiffres significatifs N en base β de ce calcul de x pour un risque de première espèce α par

$$N = \log_{\beta} \left(\frac{\overline{X}}{e_3} \right)$$

avec $e_3 = \frac{t_\alpha}{\sqrt{3}} * \sigma$, où t_α est la valeur du test de Student pour le risque α.

Cette méthode est la première méthode empirique de contrôle de l'erreur basée sur l'utilisation d'une arithmétique stochastique. Deux types d'implantations de cette méthode existent maintenant. Les implantations asynchrones qui consistent à calculer en série trois échantillons du résultat du calcul à valider et à estimer le nombre de chiffres significatifs avec la formule donnée ci dessus. Les implantations synchrones [Fay] proposées depuis peu, qui consistent à calculer en parallèle les trois échantillons du résultat et à s'arrêter dès que l'estimation du nombre de chiffres significatifs d'un résultat intermédiaire donne 0 [Vig3]. Le principe des nouvelles implantations est de n'estimer la qualité des résultats que si tous les résultats intermédiaires sont valides.

C-3-2-1-2 Modèle de Chesneaux.

Chesneaux [Che1] adapte le modèle probabiliste de Hull et Swenson [HS] à l'arithmétique stochastique de Vignes. Comme eux, il modélise les erreurs d'arrondi élémentaires par des variables aléatoires uniformes, sans biais et indépendantes entre elles. Dans le cas d'un algorithme fini, il suppose alors :

- H_1) Le nombre d'arrondis est assez grand (c'est un modèle asymptotique).
- H_2) Toutes les opérations canoniques sont analytiques (En fait, il suppose qu'on utilise uniquement les quatres opérateurs de base : multiplication, addition, soustraction et division).
- H_3) La séquence des exposants des nombres apparaissant pendant le calcul est inchangée par perturbation.

L'hypothèse H_1 est très classique dans les modèles d'erreur (erreurs de mesures, erreurs de modélisation) [Poi]. L'idée qui la sous tend est que l'inexactitude (ici due au calcul en arithmétique arrondie) sur la solution est la résultante d'un grand nombre d'erreurs élémentaires de même type. On invoque ensuite généralement le théorème de la limite centrale pour justifier un modèle Gaussien pour le résultat perturbé.

Les hypothèses H_2 et H_3 visent justement à se placer dans les conditions d'application de ce théorème. L'hypothèse H_2 est aussi classique dans les analyses de perturbations. Elle est techniquement très pratique puisqu'elle permet un développement à l'ordre un de la solution en fonction des perturbations (ici les erreurs d'arrondi locales).

L'hypothèse H_3 est plus spécifique mais elle est tout à fait nécessaire dans la théorie de Chesneaux. Elle permet en effet d'exprimer les erreurs d'arrondi locales comme toutes représentantes à une constante près de la même loi uniforme sur $[2^{-L}, 2^{+L}]$ où L est la longueur des mantisses. On note qu'elle implique qu'on ne cherche pas à calculer 0 et qu'on ne se place jamais au voisinage d'une puissance de deux, ce qui est assez restrictif. Nous avons essayé de nous en affranchir, en conditionnant par les valeurs des exposants, mais sans succès.

En développant à l'ordre 1 de X par rapport aux erreurs d'arrondi, Chesneaux montre ensuite que :

- X a pour moyenne x.
- X est approximativement Gaussienne.
- Le test de Student s'applique à X.

Cela justifie la méthode CESTAC pour certains algorithmes finis. Cette méthode bien que satisfaisante dans un très grand nombre de cas pratiques [DLLT], ne peut cependant pas être, à notre avis, considérée comme une méthode générale toujours théoriquement justifiée d'évaluation de l'erreur. Les hypothèses du modèle de Chesneaux ne sont en effet pas toujours vérifiées. Ainsi dans l'exemple ci-dessous l'estimateur du nombre de chiffres significatifs déduit du modèle de Chesneaux ne donne pas le nombre de chiffres significatifs calculé sur le résultat exact.

Exemple.

On reprend l'exemple de la suite récurrente donné en introduction :

$$a_{n+1} = 111 - \frac{1130}{a_n} + \frac{3000}{a_n a_{n-1}}$$

$$a_0 = 11/2, \quad a_1 = 61/11$$

On travaille sur un SUN 3 en double précision IEEE et en arrondi au plus près, et on étudie le résultat du calcul du 30 ième itéré de cette suite.

On rappelle que le résultat exact x vaut 5.99999989... En supposant les hypothèses du modèle de Chesneaux satisfaites, on estime par contre le résultat exact avec trois échantillons du calcul perturbé par 99.3183076878184 et on estime que le nombre de chiffres décimaux significatifs ainsi donnés sur le résultat exact est de 15. L'utilisation du modèle de Chesneaux conduit donc ici à donner pour juste un résultat complètement faux. Par suite l'une des hypothèses de ce modèle est invalidée par l'exemple et la méthode CESTAC asynchrone est mise en défaut.

C'est pour essayer d'autovalider l'estimation du nombre de chiffres significatifs par CESTAC, que l'équipe de Vignes préconise maintenant d'utiliser des implantations synchrones de cette méthode [Fay]. Cette implantation a permis pour jusqu'à présent de détecter les cas où l'estimateur du nombre de chiffres significatifs de la méthode asynchrone ne serait pas valide. C'est en particulier le cas de notre contre exemple.

Les hypothèses du modèle de Chesneaux justifiant la méthode CESTAC peuvent donc être mises en défaut. Même si les techniques de démonstration employées par Chesnaux nous semblent très intéressantes, certains arguments nous paraissent pouvoir limiter l'application pratique du modèle des perturbations qu'il propose et en particulier :

- Le fait que ce modèle ne prenne pas en compte les pertes discontinues d'information, comme les absorptions.
- La difficulté de justifier l' invariance des exposants par perturbation.
- L'utilisation d'un développement d'ordre un.
- Le fait que, comme nous le verrons, la démonstration de l'inexistence du biais entre X et x ne résiste pas à des biais locaux (pourtant existants). Ceci pourrait avoir de graves conséquences pour de grandes tailles de calculs.

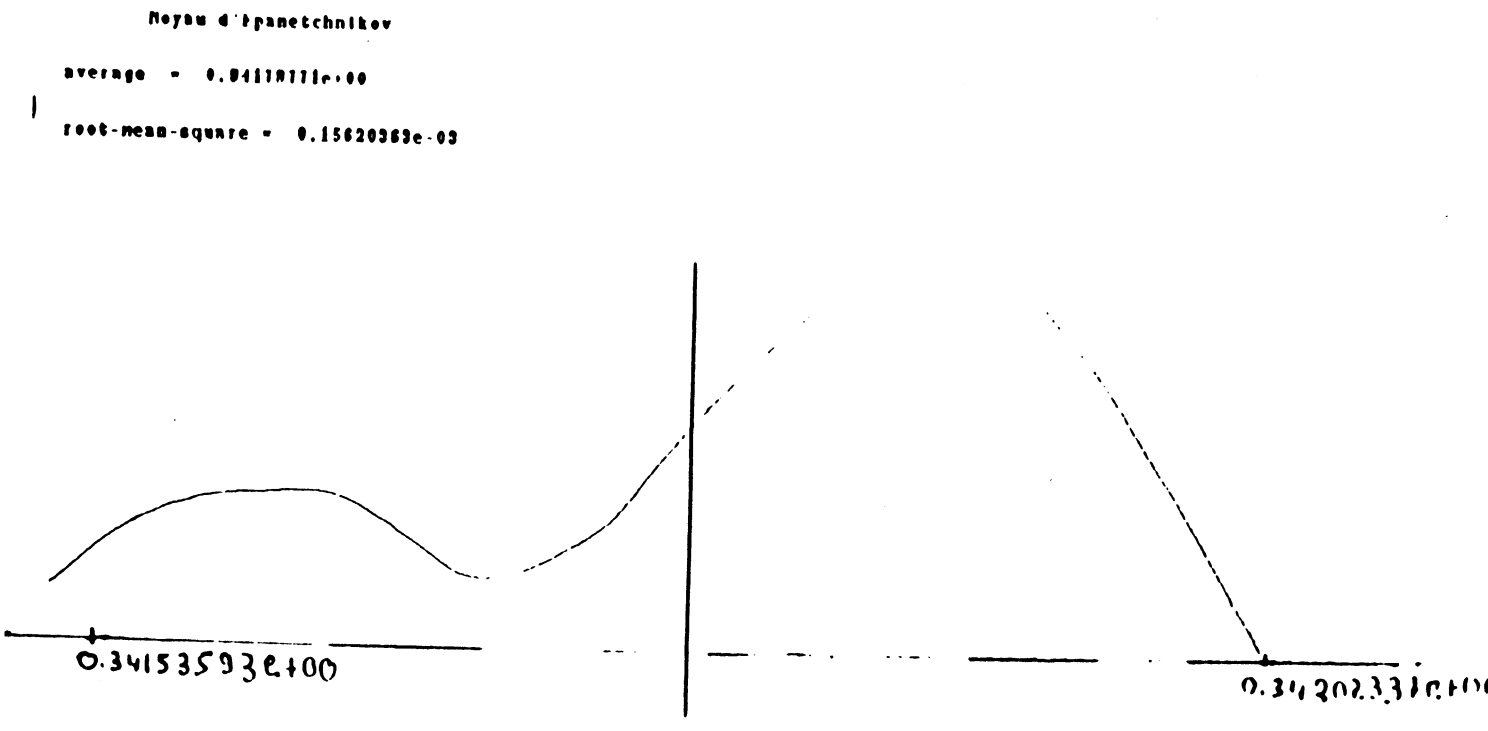
Nous donnons d'abord deux nouveaux exemples pour illustrer le rôle des non linéarités et des spécificités de l'arithmétique. On notera toutefois que dans ces exemples l'estimateur de CESTAC n'est pas mis en défaut.

Premier exemple : Rôle des non linéarités

On considère la suite non linéaire de Feigenbaum

$$x_{n+1} = 3.57 * x_n * (1-x_n),\ n = 1...499,\ x_0=0.$$

La densité de x_{500} calculée en arithmétique perturbée (avec la perturbation de Vignes pour l'arrondi au plus près) visualisée ci dessous n'est pas du tout gaussienne.



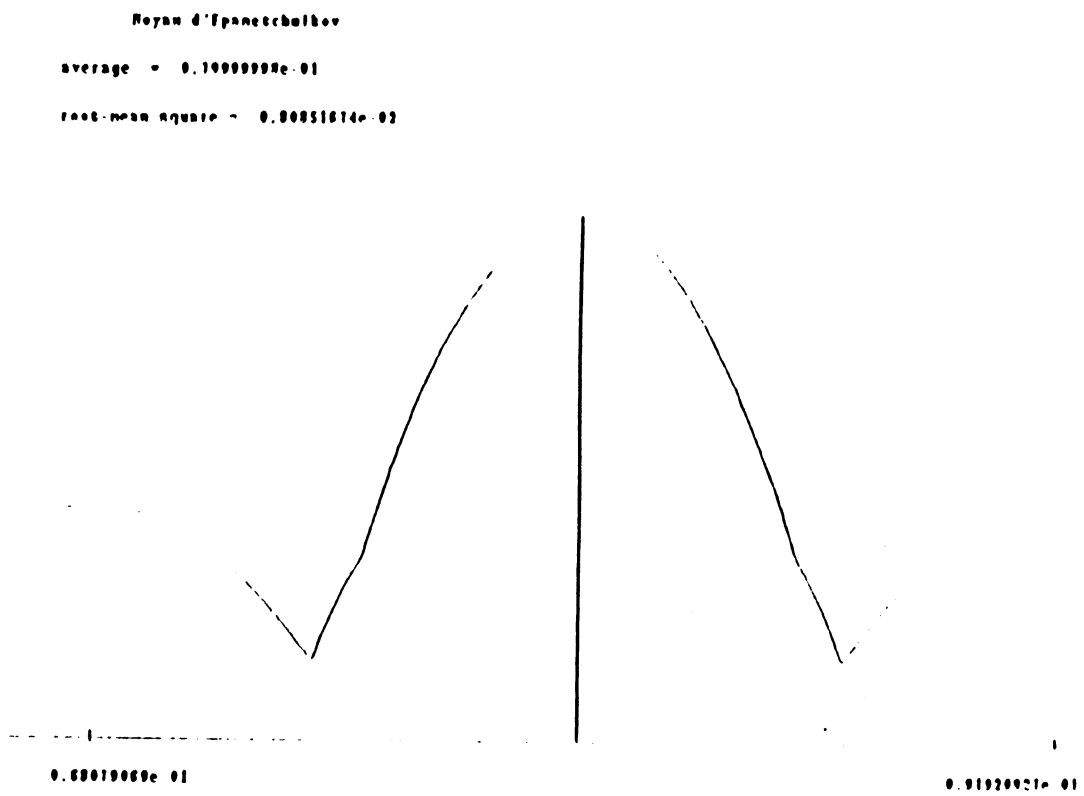
La non normalité du résultat peut peut-être s'expliquer par l'existence de fortes non-linéarités dans le calcul. Quoi qu'il en soit, le théorème central limite ou un de ses raffinements ne semblent pas pouvoir être invoqués ici pour expliquer le résultat obtenu en arithmétique perturbée.

Deuxième exemple.

On considère maintenant une simple sommation de série géométrique convergente par ordre croissant des termes :

$$\frac{1}{3^{50}}+\frac{1}{3^{49}}+\ldots+\frac{1}{3}+1$$

En appliquant la perturbation de Vignes, seule la dernière perturbation effectuée est visible. En effet, les autres perturbations disparaissent par absorption lors des additions en virgule flottante. Par suite comme le montre l'histogramme ci dessous, la densité du résultat est purement discrète (à trois pics).



Cet exemple montre bien comment les spécificités de l'arithmétique peuvent empêcher de pouvoir faire l'hypothèse de Chesneaux selon laquelle un grand nombre de perturbations influencent le résultat .

Problème de biais.

Nous abordons maintenant un argument plus théorique, à savoir l'instabilité possible du modèle par rapport à des biais locaux. Le problème est tout à fait identique à celui des modèles probabilistes. Chesneaux suppose :

$$X=x+\sum_{i=0}^{N} a_i.e_i$$

Où N est le nombre d'arrondis effectués, e_i la mesure aléatoire du ième arrondi et a_i une constante ne dépendant que du calcul. Il suppose de plus que e_i est d'espérance nulle.

Revenons d'abord sur cette hypothèse avant d'examiner ses conséquences. Nous allons voir que l'on peut observer parfois des biais locaux de l'ordre du bit de garde (c'est à dire des espérances de e_i non nulles) :

• Intéressons nous au calcul de la somme de 2 nombres x et y positifs, milieux d'intervalles machine en troncature perturbée par la perturbation de Vignes. Nous noterons :

- x_- le nombre machine prédécesseur de x, et x_+ le nombre machine successeur de x.
- exp (z) l'exposant de z dans sa représentation virgule flottante en base 2.
- S (T) le support de la variable aléatoire T.
- $\mu(x)$ la variable aléatoire qui vaut x_+ avec une probabilité $\frac{1}{2}$ et x_- avec une probabilité $\frac{1}{2}$.
- $\oplus$ la somme tronquée.
- n le nombre de chiffres de mantisse
- $I_X = [x_-, x_+]$, et $I_Y = [y_-, y_+]$ les intervalles machine contenant x et y.
- $X = \mu(x)$, $Y = \mu(y)$ les représentations de x et y par la méthode de Vignes en troncature.
- $\oplus_V$ la somme perturbée de x et y. $\oplus_V$ est définie par $x \oplus_V y = \mu(X \oplus Y)$

• Nous essayons de vérifier le modèle de Chesneaux de la somme $\oplus_V$ au niveau du biais local. Ce modèle se base d'abord sur le modèle de Feldstein : Pour tout I intervalle machine du support de X+Y, $(X+Y)|_I$ est une variable aléatoire uniforme sur I.

Or pour toute loi U_I uniforme sur I intervalle machine, $E(U_I)$ est égal $E(x \oplus_V y\,|_I)$. Par conséquent : $E(x \oplus_V y\,|_I) = E(X+Y|_I)$

Il vient alors la propriété d'absence de biais local au niveau de l'arrondi de la somme :

$$E(x \oplus_V y) = E(X+Y)$$

qui implique :

$$E\,(X+Y) = E\,(\Sigma U_I \,.\mathrm{prob}(X+Y \in I)$$

Examinons cette propriété dans le cas particulier où $x \oplus y_+ = y_+$ (ce qui se produit lorsque y est grand devant x)

<u>CAS I</u> : $x_+ \oplus y_- = y_+$

a) $x_- \oplus y_- = y_+$

Alors S (X+Y) = $[y_-, y_+]$ et on a :
$E(X \oplus_V Y) = (y_{+_+} + y_+)/2 = y_+ + 2^{\exp(y_+)-L-1}$
$E(X+Y) = y_+ + x_+ - 2^{\exp(y_+)-2L-2} - 2^{\exp(y_+)-L-1}$
d'où un biais $b = -x_+ + 2^{\exp(y_+)-L} + 2^{\exp(y_+)-2L-2} \leq 2^{e(y_+)-L-1}$ (atteint)

Dans ce cas, on a donc un biais sur le résultat de l'ordre du bit de garde. La moyenne n'est ni l'arrondi au plus près du vrai résultat, ni le vrai résultat, ni le vrai résultat tronqué...

b) $x_{.} \oplus y_{.} = y_{.}$

$$E(X+Y) = y_{+} + x_{+} - 2^{exp(y_{+})-2L-2} - 2^{exp(y_{+})-L-1}$$
$$S(X+Y) = [y_{.}, y_{+_{+}}]$$
$$E(X \oplus_{V} Y) = E(1/4 * U[y_{.}, y_{+}] + 3/4 * U[y_{+}, y_{+_{+}}]) = y_{+} + 2^{exp(y+)-L-2}.$$

Là aussi, on a un biais. Celui ci peut être au maximum de $2^{exp(y_{+})-L-2} + 2^{exp(y_{+})-2L-2}$ c'est à dire de l'ordre du deuxième bit de garde.

Cas II : $x_{+} \oplus y_{.} = y_{.}$

Dans ce cas,

$$E(X+Y) = y_{+} + x_{+} - 2^{exp(x_{+})-L-1} - 2^{exp(y_{+})-L-1}$$
$$E(X \oplus_{V} Y) = E(1/2 * U[y_{.}, y_{+}] + 1/2 * U[y_{+}, y_{+_{+}}]) = y_{+}.$$

On a donc un biais de l'ordre du bit de garde (au maximum).

Conclusion.

La somme de nombres en arithmétique perturbée peut donc conduire à des biais locaux de l'ordre du bit de garde.

Remarque : Une deuxième raison plus pratique peut aussi expliquer l'existence de biais locaux, c'est tout simplement l'imperfection inhérente à tout générateur aléatoire.

Conséquence sur l'applicabilité de l'estimateur de Chesneaux pour de grands calculs.

Comme on s'y attendait, dans le développement :

$$X = x + \sum_{i=0}^{N} a_{i} \cdot x_{i}$$

On doit donc supposer $E(e_{i}) = m_{i} \neq 0$. Qu'en est t-il alors de la validité de l'estimation de l'erreur proposée par Chesneaux ? Si toutes les autres hypothèses de Chesneaux restent vraies la convergence de X vers une loi normale $N(m,\sigma)$ (et donc l'applicabilité du test de Student à X dès que N est suffisament grand) est encore démontrable. Cependant, la moyenne m vaut maintenant :

$$x + \sum_{i=0}^{N} a_{i} m_{i}$$

C'est à dire qu'il existe un biais entre le résultat exact x et le résultat moyen m. L'estimation du nombre de chiffres significatifs par la formule utilisée dans CESTAC donne donc, *a priori*, seulement une estimation du nombre de chiffres significatifs donnés sur le résultat moyen et pas sur le résultat exact. C'est pourquoi Chesneaux [Che2] et Chatelin [Cha2] ont étudié (sous l'hypothèse d'une loi normale) la validité de cet estimateur du seul nombre de chiffres significatifs sur m comme estimateur du nombre de chiffres significatifs sur le résultat exact. Ils concluent tous deux que pour des biais (x-m) inférieurs à 21 fois l'écart-type l'estimateur de CESTAC au seuil α est toujours valide.

Mais l'étude de Chatelin [Cha2] souligne que pour un biais supérieur à 21 fois l'écart-type la fiabilité de CESTAC chute à 1-α. La plupart des problèmes de laboratoire (donc de taille assez petite) que nous avons traités ont un biais inférieur à cette borne critique de 21σ. Cependant, Chatelin [Cha3] a donné l'exemple de l'algorithme :

```
x:=10^6
y:=10^-3
s:=0
Pour i = 1 à n faire
        s=s+ ((x+ y -x )/y)
finfaire
```

pour lequel le rapport biais sur écart type croit avec la taille n du calcul jusqu'à être au delà de cette borne. Nous avons également rencontré divers exemples d'augmentation du rapport "biais sur écart type" avec la taille des problèmes au cours de notre étude de l'inversion d'un système linéaire par l'algorithme de GAUSS [Fra]. Nous essayons maintenant de préciser ce phénomène pour une classe d'algorithmes :

On se place dans l'ensemble des algorithmes pour lesquels pour tout i :

1) $a_i\ m_i > m$
2) $|a_i| < a$

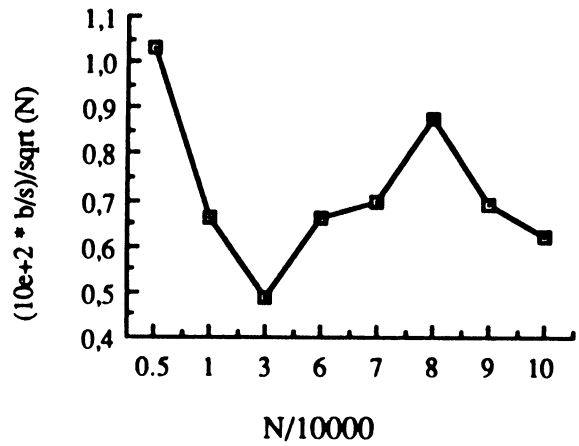
En conservant les autres hypothèses de Chesneaux, on obtient une majoration de la variance du résultat perturbé de la forme $k\sqrt{N}$ (k constante fixée). Pour le biais E(X)-x on obtient par contre un minorant de la forme k'N. On minore ainsi le rapport biais sur variance par une fonction de la forme $K\sqrt{N}$.

Cette croissance du rapport biais/écart-type en $\sqrt{N}$ où N est le nombre de perturbations (ou au moins en la taille des problèmes traités pour un même type de calcul) semble assez générale (cf à ce sujet le paragraphe sur les modèles probabilistes). Elle peut sans doute se prouver (en conservant bien sûr les autres hypothèses du modèle de Chesneaux) pour d'autres classes d'algorithmes que celle étudiée ici. De plus, elle constitue souvent une réalité expérimentale. On donne par exemple ci-dessous un exemple d'évolution de la performance de l'estimateur "variance" en $1/\sqrt{N}$ probablement imputable au générateur aléatoire.

Exemple.

Notre exemple consiste en l'étude de la sommation par ordre des termes décroissant de N termes de la série de terme général $\frac{1}{2n}$. Si le générateur aléatoire était parfait, le biais b sur la Nième somme partielle serait au plus en logN et donc le rapport biais sur variance en $\frac{\log N}{\sqrt{N}}$. Pourtant, comme le montre le diagramme suivant, on observe une évolution de ce rapport proche de $\sqrt{N}$ comme si à chaque étape de calcul on avait un biais de l'ordre du résultat.

Effet d'accumulation de biais répétitifs



Ainsi, l'augmentation du rapport biais sur variance avec la taille des problèmes pourrait-elle constituer une limite à CESTAC. Nous n'avons évidemment fait que mettre en évidence une tendance. Nous conjecturons cependant que l'étude de CESTAC sur des problèmes de très grande taille pourrait préciser cette limite.

C-3-2-2 PRECIX

Ayant observé ce problème de biais, M.C. Brunet et F. Chatelin proposent quant à elles un modèle différent, PRECIX [CB] [Bru], pour la résolution d'un système linéaire par l'algorithme de Gauss et le calcul d'une valeur propre par l'algorithme QR. Contrairement à celui de Chesneaux, ce modèle ne fait aucune hypothèse sur les erreurs d'arrondi locales et spécifie seulement l'effet macroscopique des arrondis à l'aide d'une analyse rétrograde à la Wilkinson (cf chapitre III). Son application suppose :

H_1) Que le nombre d'arrondis est "grand" et que l'algorithme perturbé résout le problème exact pour une perturbation normale des données.

H_2) Que les perturbations sur deux données distinctes sont indépendantes.

Sous ces hypothèses MC Brunet et F Chatelin [Bru] infèrent alors un modèle paramétrique pour la solution perturbée. Dans le cas de l'algorithme de Gauss, elles déduisent par exemple que la solution perturbée suit une loi normale.

Ce modèle est historiquement très intéressant, car il est le premier à prendre en compte la possibilité de biais entre le résultat perturbé X et le résultat exact. Nous donnons cependant un exemple de résolution linéaire par l'algorithme de GAUSS pour lequel PRECIX est invalidé.

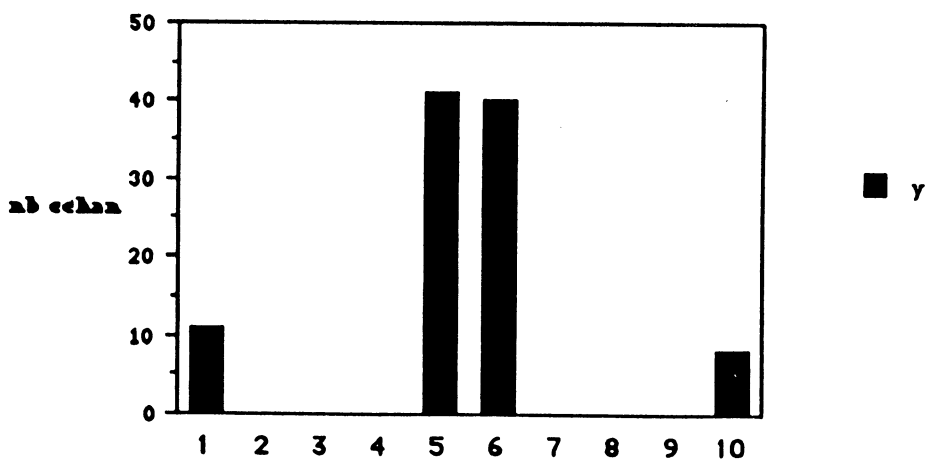
Exemple.
On travaille sur un SUN 3 en double précision IEEE et en arrondi au plus près, et on étudie la perturbation de l'algorithme de Gauss sans pivot pour la matrice de taille n suivante :

$$T(e,n)=\begin{bmatrix} 1 & 0 & 0 & \dots & & 0 & 0 \\ 1 & 4 & 0 & 0 & & & 0 \\ . & & & & \dots & & \\ . & .... & i*j... & i^2 & & & \\ . & & & & & 0 & \\ 1 & 2*(n-1) & & & & (n-1)^2 & 0 \\ 1-\frac{1}{e} & 2*n & . & . & . & n*(n-1) & \frac{1}{e^2} \end{bmatrix}$$

On s'intéresse plus particulièrement à la solution du système $T(e, n).s = u_n$ où u_n est le vecteur de taille n qui a toutes ses composantes égales à 1, et spécialement à la $n^{ième}$ composante s_n de s qui est égale à e.

Dans notre expérience, le paramètre e vaut 2^{40} (approximativement 10^{12}), on observe 100 échantillons de la dernière composante de la solution du système pour une taille n = 10 et on effectue la perturbation de Vignes de l'algorithme. On en trace l'histogramme. On note en abscisse le numéro de la classe et en ordonnée le nombre d'échantillons lui correspondant. Les échantillons sont séparés en 10 classes équidistantes suivant des valeurs croissantes.

Cas où un modèle normal ne convient pas



• **Résultats** :

La distribution du résultat perturbé n'est visiblement pas Gaussienne. Plus exactement, il semble qu'on ait une loi à 3 pics correspondant aux valeurs suivantes :

- Valeur du premier pic : -4. 19e+6
- Valeur du deuxième pic : Valeurs comprises entres -5e+5 et 5.e+5
- Valeur du troisième pic : 4. 19e+6

Cela montre donc que PRECIX n'est pas toujours valide. On note par ailleurs sur cet exemple que l'arithmétique non perturbée est parfaite et que bien que le biais soit très grand devant l'écart-type (environ 10^6 fois plus) la méthode CESTAC détecte dans ce cas la non validité de son estimation du résultat exact. Nous déconseillons donc toute utilisation aveugle de PRECIX. Nous pouvons également noter sur PRECIX :

• Qu'il s'agit aussi d'un modèle asymptotique (sur la précision) basé sur un développement à l'ordre minimal (de la fonction des données que cherche à calculer l'algorithme).
• Que ce modèle fait implicitement référence au théorème central limite (Hypothèse H_1) et donc que son utilisation suppose qu'un nombre suffisant de perturbations agit effectivement sur le résultat.

Nous soulignons enfin rapidement un problème lié à l'utilisation pratique du type de démonstration de MC Brunet. Dans [Bru] la normalité du résultat X pour l'algorithme de Gauss de résolution du système Ax = b se déduit de celle du résidu Ax-b en utilisant le simple fait qu'une transformation linéaire d'un vecteur Gaussien est Gaussienne. Cela ne doit cependant pas conduire à tester pratiquement le modèle PRECIX en regardant le seul résidu car bien qu'une transformation linéaire d'une variable Gaussienne soit Gaussienne, une transformation linéaire d'une loi éloignée d'une loi normale peut très bien être approximativement normale.

C-3-3 Problèmes.

Ainsi, l'effet d'une arithmétique stochastique virgule flottante est difficile à interpréter systématiquement. Il dépend trop du type de problème à résoudre et des valeurs numériques des données. Tous les modèles existants ont des contre exemples. Bien sûr, ceux que nous donnons ne sont que des cas d'écoles mais nous conjecturons que les phénomènes évoqués dans ces exemples pourraient se rencontrer dans des cas pratiques. Par ailleurs, nous soulignons que nous ne connaissons pas actuellement les domaines de validité de ces modèles.

Plusieurs raisons peuvent à notre avis expliquer ces difficultés :

• Le comportement complexe des erreurs d'arrondi locales en arithmétique flottante :

- elles ne sont ni absolues ni relatives.
- la probabilité de ne faire aucune erreur d'arrondi à une étape n'est pas négligeable [KL1].
- elles sont biaisées [MM]

• Les trop fortes non-linéarités parfois associées aux instabilités invalident les développements à l'ordre un.

• La manipulation de nombres de différents ordres de grandeur et donc d'erreurs d'arrondi de différents ordres de grandeur peut empêcher d'appliquer les théorèmes asymptotiques (car alors seuls quelques arrondis influent sur le résultat final).

• Enfin l'existence de phénomènes de singularisation "mangeurs de perturbations" peut empêcher l'application de théorèmes asymptotiques.

Beaucoup de ces difficultés de la modélisation probabiliste avaient été déjà évoquées par Hull et Swenson [HS] à propos de leur modèle probabiliste.

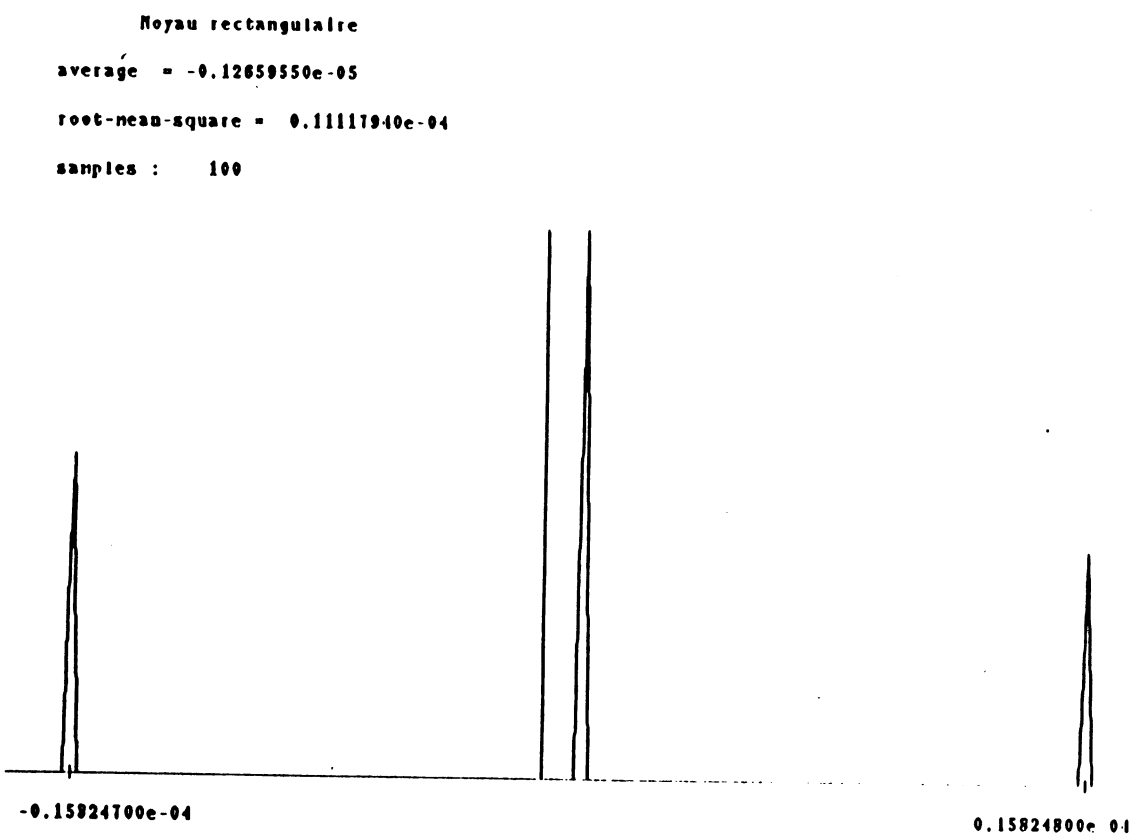
Une approche paramétrique générale des arithmétiques flottantes perturbées parait irréaliste. Montrer des résultats du type de ceux de Chesneaux mais avec des hypothèses plus générales, ou tout au moins plus facilement vérifiables nous semble difficile. On peut simplement espérer adapter les résultats d'autres approches (le modèle d' Henrici par exemple) dans certains cas.

Au contraire, un modèle comme celui de Chesneaux [Che1] pourrait sembler plus intéressant pour d'autres arithmétiques (fixe, logarithmique) où les problèmes ne semblent pas si importants tout au moins si on se limite à quelques opérateurs.

Exemples.

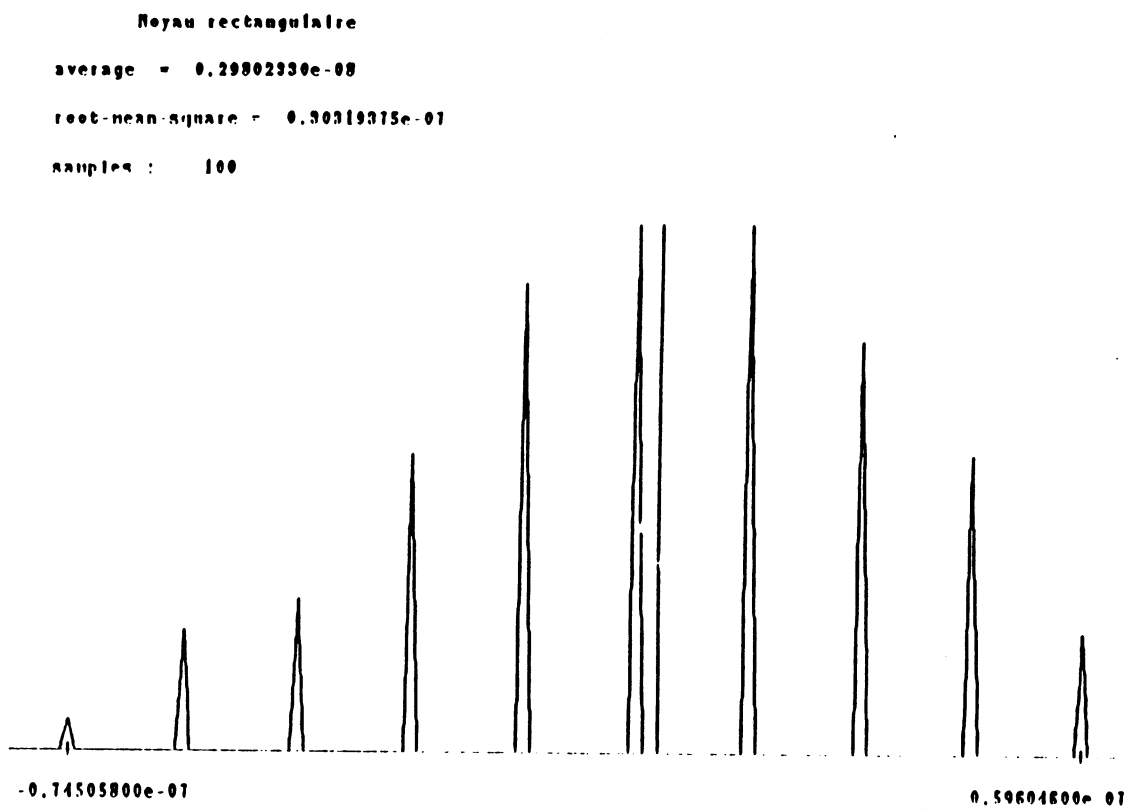
Nous reprenons l'exemple de la série géométrique en arithmétique fixe. La densité du résultat perturbé se rapproche ici bien d'une Gaussienne car on est certain que toutes les perturbations influencent le calcul.

Histogramme de la sommation de la série géométrique en arithmétique fixe perturbée.



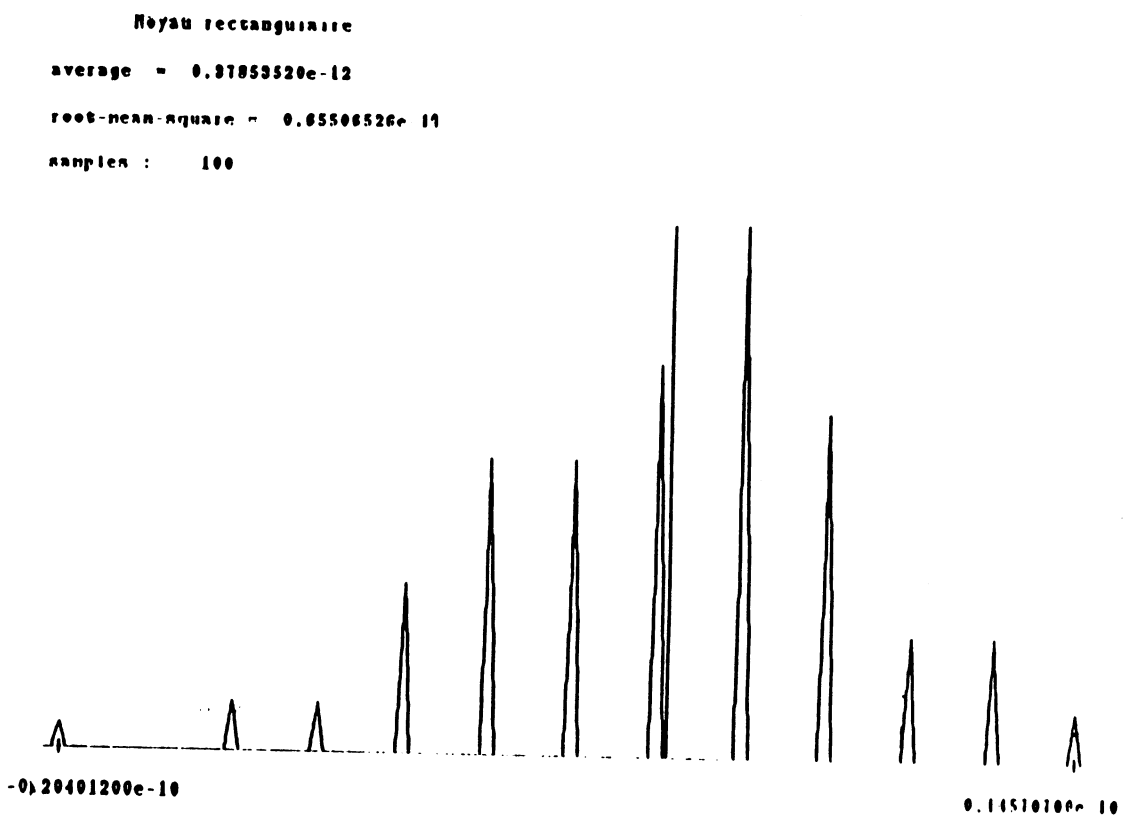
De la même manière l'arithmétique logarithmique se comportera bien pour les multiplications ainsi que le suggère l'exemple suivant du calcul du produit des 1/i en arithmétique perturbée.

Histogramme du produit des 1/i en arithmétique logarithmique perturbée



Mais là encore, comme le souligne l'étude de la série géométrique en arithmétique logarithmique, il semble peu probable de trouver une loi générale.

Histogramme de la sommation de la série géométrique en arithmétique logarithmique perturbée.



C-4 Sur l'utilisation de l'approche statistique.

Nous venons de présenter trois approches statistiques différentes de l'erreur d'arrondi : les modèles globaux d'évaluation des performances ([Knu], [FG]), les modèles probabilistes d'un calcul ([Hen1], [HS]) et les méthodes de perturbation des calculs ([PV2], [Cha3]). Ces trois approches apparaissent séparément dans la littérature et le lien est rarement fait entre elles. En conclusion, on essaie d'abord de réexaminer rapidement ces méthodes, en montrant dans quel sens elles peuvent ou ne peuvent pas se fonder mutuellement.

C-4-1 Lien entre les trois approches.

Même si elles sont *a priori* peu reliées, les trois approches probabilistes de la propagation des erreurs d'arrondi sont quelquefois utilisées ensemble. Les modèles globaux [Knu], [FG] représentent simplement un comportement moyen des erreurs d'arrondi dans un ensemble de calculs. Mais bien qu'ils ne représentent pas l'arrondi dans un calcul donné ils servent de base à la détermination de beaucoup de paramètres dans des modèles probabilistes d'un calcul. ([Che1], [Hen2]). Ce choix se justifie empiriquement mais n'a que peu de bases théoriques. C'est pourquoi, il nous semble difficilement justifiable d'en inférer des propriétés mathématiques. Au contraire, les modèles probabilistes d'un calcul nous semblent parfois pouvoir fonder une interprétation de l'effet de perturbations aléatoires d'un calcul. Le modèle de Hull et Swenson [HS] sert déjà de référence aux travaux de Chesnaux [Che1]. Le modèle d'Henrici [Hen1], [Hen2] quant à lui n'a pas encore été adapté aux arithmétiques perturbées. Cette adaptation pourrait peut-être se révéler instructive. Inversement, l'arithmétique perturbée semble rendre les modèles probabilistes "opérationnels". Elle permet en particulier de rendre accessibles les paramètres du modèle. Par ailleurs même, elle rend plus réalistes, tout au moins en théorie, les hypothèses de non corrélation.

C-4-2 Qu'attendre de ces modèles ?

Bien que déjà ancienne, l'approche statistique des arrondis est encore peu utilisée en pratique pour l'analyse de la qualité des programmes. Comme nous le savons, la plupart des modèles classiques sont difficiles à fonder (l'aléa est difficile à expliciter ou fictif, les linéarisations et les non corrélations sont difficiles à justifier pratiquement [Cha3], [Fra], [Knu]) ou non effectifs [Hen1], [HS]. C'est pourquoi, pour contrôler quantitativement l'erreur arithmétique, il est compréhensible que, malgré ses défauts, l'on préfère utiliser une arithmétique d'intervalles [KM], qui a des fondements simples, plutôt que des méthodes de perturbations. Par contre, nous pensons qu'une approche statistique pourrait être utilisée plus souvent pour extraire des informations qualitatives sur les effets des arrondis. Un modèle d'inférence statistique est en effet intrinsèquement plus informatif qu'un modèle déterministe. Les méthodes de sondage par perturbations nous paraissent pouvoir être particulièrement intéressantes, en particulier par la possibilité de les mettre en oeuvre systématiquement sur n'importe quel type d'algorithmes. Il faut poursuivre leur formulation en revenant peut-être à leurs principes génériques. Nous suivons là F. Chatelin qui a déjà entamé une telle utilisation [Cha4].

D - CONCLUSION.

Les principes qui régissent les différentes méthodes de contrôle d'erreur existantes sont donc très variés. Il peut s'agir d'encadrements déterministes des résultats (méthodes fondées sur des arithmétiques d'intervalles), de majorations récursives, d'estimations probabilistes *a priori* ou basées sur l'utilisation d'arithmétiques stochastiques. En regard de cette diversité de principes, les objectifs et possibilités pratiques des différentes méthodes de contrôle d'erreur nous semblent par contre très semblables. Elles fournissent la plupart du temps une estimation de l'erreur arithmétique plus ou moins fiable. Dans le cadre de cet objectif purement quantitatif, l'utilité pratique de la plupart des méthodes n'est pas toujours clair.

Les approches probabilistes en particulier ne nous semblent pas devoir être retenues dans le cas général. Si l'on vise simplement à obtenir une estimation fiable de l'erreur arithmétique, un bon outil d'arithmétique d'intervalles pourrait être suffisant. Mais remémorons nous un instant les différents objectifs du contrôle de l'erreur arithmétique : Evaluer, corriger, repérer, expliquer... l'erreur. Dans cette optique un outil d'arithmétique d'intervalles est trop limité. Une borne déterministe de l'erreur arithmétique n'a aucun pouvoir explicatif de l'erreur d'arrondi. Elle n'indique pas si l'erreur observée est inhérente au problème ou due à l'arithmétique, si en augmentant la précision on diminuera significativement l'erreur. Dans l'optique de l'extraction de telles informations plus qualitatives sur l'erreur arithmétique, une approche statistique et notamment l'utilisation d'arithmétiques perturbées semble pouvoir, au contraire, parce qu'elle est plus informative, retrouver tout son intérêt. Mais il importe pour cela de se baser sur une formulation claire de la stabilité arithmétique qui manque à la plupart des méthodes classiques présentées antérieurement.

BIBLIOGRAPHIE DE L'APPROCHE DETERMINISTE

[Ada] ADAMS, Enclosure methods and scientific computation. Proc of IMACS, PARIS, juillet 1988.

[Bau] FL BAUER, Computational graphs and rounding error. SIAM J of Num Anal, vol 11, N°11, mars 1974.

[BNR] J.H BLEHER, AE NOEDER, SM RUMP, ACRITH, High accuracy arithmetic, an advanced tool for numerical computation, IEEE, 1985.

[Dav] JH DAVENPORT, Survey of symbolic applications for numerical computation. Copyright by the diamond consortium, 1987.

[KL] W KAHAN et E LEBLANC, Anomalies in the IBM ACRITH package, IEEE, 1985.

[KR] E KAUCHER et SM RUMP, Generalised iteration methods for bounds of the solution of fixed point operator equation. Computing 24, pp 131-137, 1980.

[KM] U W KULISCH et WL MIRANKER, The arithmetic of the digital computer, a new approach. SIAM Rev, vol 28 n°1, mars 1986.

[Lan] E LANGE, Implementation and test of the Acrith facility in a system 370. IEEE transactions on computers, vol C26, N°9 septembre 1987.

[LS1] JI LARSON et AH SAMEH, Efficient calculation of the effects of roundoff errors. J of the ACM vol 4 N°3 pp 228-236, septembre 1978.

[LS2] JI LARSON et AH SAMEH, Algorithms for roundoff error analysis, a relative error approach. Computing 24, pp 275-297, 1980.

[Mil1] W MILLER, Computer search for numerical instability. J of the ACM, vol 22 N°4 pp 512-521, octobre 1975.

[Mil2] W MILLER, Round off analysis by direct comparaison of two algorithms. SIAM J of Num Anal, vol 13 N°3, juin 1986.

[MBC] M MUTRIE, R BARTELS, B CHAR, Floating point error analysis using symbolic algebraic computation. University of Waterloo, Dep of computer science, Symbolic comp group, mai 1986.

[Rum] S RUMP, Algebraic computation, numerical computation and verified inclusion. Trends in computer algebra, proceedings, mai 1987.

[Stu] F STUMMEL, Strict a posteriori error and residual bounds for gaussian elimination in floating point arithmetic. Computing 37, pp 103-137, 1980.

[Tsa] N KUAN TSAO, A simple approach to error analysis of division freee numerical algorithms, IEEE transactions on computers, vol C32, N°4, avril 1983.

[Zad] P ZADUNAISKY, On the estimation of errors propagated in the numerical integration of ordinary differential equations. Numer Math 22, pp 21-39, 1976.

BIBLIOGRAPHIE DES APPROCHES STATISTIQUES

[BR] BENSHOP NF, HC RATZ, A mean square estimate of the generated round off error in constant matrix iterative processes. J of the ACM, vol 19, n°2, pp 48-62, 1971.

[BF] D BLOYET et P FURON, Incertitudes de calcul dans un processeur de fourier rapide : modélisation et expérience. Traitement du signal, vol 5, n°1, 1988.

[Bre] R P BRENT, On the precision attainable with various floating-point number systems. IEEE transactions on computers, vol C22, N°6, 1973.

[Bru] MC BRUNET, Contribution à la fiabilité de logiciels numériques et à l'analyse de leur comportement : une approche statistique. Thèse de l'université PARIS 9, janvier 1989

[Buc] W BUCHHOLZ, Planning a computer, Project stretch, Mc Graw Hill, New York, 1962.

[CB] F CHATELIN et MC BRUNET, A probabilistic round off error propagation model, application to the eigenvalue problem. To appear in Reliable Numerical Computation, 1988.

[Cha1] F CHATELIN, On some problems raised by the use of probabilistic models for round-off error and precision control in scientific computing. IBM, Centre scientifique de Paris, étude F.112, mars 1987.

[Cha2] F CHATELIN, Sur le taux de fiabilité de la méthode CESTAC. Note CRAS n° 264, 1988.

[Cha3] F CHATELIN, Analyse statistique de la qualité numérique et arithmétique de la résolution approchée d'équations par calcul sur ordinateur, Etude F.133, centre scientifique IBM FRANCE, avril 1988.

[Cha4] F CHATELIN, Résolution approchée d'équations sur ordinateurs. Notes de cours de DEA de statistique, Université de Paris 6, 1989-1990.

[Che1] J M CHESNEAUX, Etude théorique et implantation en ADA de la méthode CESTAC. Thèse de l'université PARIS 6, Avril 1988.

[Che2] J M CHESNEAUX, Sur la robustesse de la méthode CESTAC. Note CRAS, 1988.

[Cod] WJ CODY, Static and dynamic numerical characteristics of floating-point arithmetic. IEEE transactions on computers, vol C22, n°6, pp 598-601, 1973.

[DLLT] I DUFF, J LAMINIE, A LICHNEWSKY, F THOMASSET, An experiment with arithmetic precision in linear algebra computations. Rapport de recherche, INRIA, 1986.

[FV] JP FAYE et J VIGNES, Stochastic approach of the permutation-perturbation method for round off error analysis. Transactions of IMACS, vol 1 N° 4, juillet 1985.

[Fay] J P FAYE, Implémentation synchrone de CESTAC, C.R.Acad.Sci.Paris, t.309, Série I, p637-640, 1989.

[FG] A FELDSTEIN et R GOODMAN, Convergence estimates for the distribution of trailing digit. J of the ACM, vol 23, n°2, pp 287-297, avril 1976.

[For] FORSYTHE. Note on rounding-off errors. SIAM Rev 1, 1966-1967.

[Fra] P FRANCOIS, Mesure de l'erreur arithmétique et perturbations des calculs. Rapport de recherche IMAG-TIM3, janvier 1989.

[Ham] RW HAMMING, On the distribution of numbers. Bell syst tech Journal, vol 49, pp 1609-1625, 1970.

[Hen1] P HENRICI, Discrete variable methods in ode. Wiley, New York, 1963.

[Hen2] P HENRICI, Test of probabilistic models for the propagation of roundoff errors. Com of the ACM, vol 9, pp 409-410, 1966.

[HS] HULL et SWENSON, Test of probabilistic models for propagation of roundoff error. Com of the ACM, Vol 9 N° 2, 1966.

[KL1] T KANEKO et B LIU, On local roundoff errors in floating-point arithmetic. J of the ACM, vol 20, n°3, pp 391-398, 1973.

[KL2] T KANEKO et B LIU, Accumulation of roundoff error in fast fourier transforms. J of the ACM, vol 20, n°3, pp 391-398, 1973.

[Knu] KNUTH, The art of computer programming, vol 2. Addison wesley edition, 1973.

[MM] J MARASA et D MATULA, A simulative study of correlated error propagation in various finite precision arithmetics. IEEE transactions on computers, vol C22, N°6, juin 1973.

[Poi] H POINCARE, La Science et l'hypothèse. Editions Flammarion, Paris.

[PV1] M LA PORTE et J VIGNES, Etude statistique des erreurs dans l'arithmétique des ordinateurs, application au contrôle des résultats d'algorithmes numériques. Num Math 23, pp 63-72, 1974.

[PV2] M LA PORTE et J VIGNES, Error analysis in computing. Information processing, 1974.

[Rad] HA RADEMACHER, On the accumulation of errors in processes of integrating on high speed calculating machines. Proc symp large digital calculating machinery, Harvard Comp lab series, vol 16, pp 176-187, 1948.

[RS] TG ROBERTAZZI et S C SCHWARTZ, Best ordering for floating-point addition. ACM trans on math software, vol 14, n°1, Mars 1988.

[SS] J. K SCHEIDT et C. W SCHELIN, Distribution of floating-point numbers. Computing 38, pp 315-324, 1987.

[Sch] F SCHLESINGER, On the errors in the sum of tabular quantities. Astronomical Journal, vol 30, pp 183-190, 1917.

[Tur] PR TURNER, The distribution of leading significant digits, Ima J of Num Anal, vol 2, pp 407-422, 1982.

[Vig1] J VIGNES, New methods for evaluating the validity of the results of mathematical computations. Mathematiccs and computers in simulation, Transaction of IMACS, Vol XX, N°4, pp 227-249, December 1978.

[Vig2] Mathematics and computer in simulation. Special issue on stochastic methods in roundoff error analysis, IMACS, 1988.

[Vig3] J VIGNES, Zéro mathématiques et zéro informatique. C.R. Acad. Sc. Paris, t. 303, Série I, n°20, 1986.

[Vig4] J VIGNES, Contrôle et estimation stochastique des arrondis de calcul, AFCET interfaces n° 54, Avril 87, pages 3-11.

CHAPITRE III
STABILITE ARITHMETIQUE

A - INTRODUCTION.

Comme nous l'avons déjà rappelé, la représentation machine d'un algorithme n'est qu'une approximation (ou une perturbation) du calcul exact. Nous avons vu que de nombreuses méthodologies existaient pour évaluer une borne d'erreur sur un résultat pour cette perturbation, il nous parait cependant nécessaire de disposer d'autres outils d'analyse de la qualité arithmétique d'un programme, plus généraux ou plus à même de rendre compte de ses propriétés mathématiques. Les méthodes de contrôle d'erreur sont essentiellement empiriques, nous croyons au contraire qu'il faut un outil théorique d'analyse.

Aucune théorie complète de la stabilité arithmétique n'existe dans la littérature. Cependant, de nombreux auteurs ont posé les bases d'une telle "boite à outils". Les concepts de conditionnement [Cha1], [Ric], [Gol], [Dem], [Fle] par exemple qui permettent d'analyser l'effet d'une erreur sur les données lors de la résolution d'un problème donnent déjà un fondement mathématique solide à une mesure de la stabilité. L'analyse régressive des erreurs de Winkilson [Win], les notions empiriques de stabilité inverse et directe ou surtout la théorie de la stabilité de Chatelin [Cha2] (pour des problèmes du type $F(X) = 0$) sont également des outils intéressants. Par ailleurs, de nombreux concepts existent pour traiter des approximations numériques comme les discrétisations d'équations différentielles ou pour étudier le comportement d'itérations. Ainsi la notion d'ordre de convergence donne un indice de qualité d'une telle approximation. Il parait évidemment utile d'essayer d'adapter ces outils au cas de la discrétisation machine, pour laquelle le problème mathématique n'est pas très différent.

Dans ce chapitre, on part des travaux cités précédemment pour donner ensuite des outils généraux de mesure de la stabilité arithmétique.

B - GENERALITES.

B-1 Modèle d'algorithme.

Dès le premier chapitre, nous avons représenté un algorithme numérique par un système dynamique sur $\mathbb{R}$. Nous continuerons ici de travailler sur un tel modèle en représentant tout algorithme numérique A par un système dynamique stochastique du type :

$$x_{t+1} = f_t(x_t),\ t = 0 \ldots T\text{-}1$$
$$x_0 \text{ donné}$$

où x_i est un vecteur aléatoire sur $\mathbb{R}^{v(i)}$; T et v(i) des entiers; les f_t des fonctions aléatoires.

Par la suite, la $j^{ème}$ composante de x_i sera notée x_i^j; $\mathfrak{A}(T, v(0) \ldots v(T))$ désignera l'ensemble des algorithmes à T étapes où la $i^{ème}$ étape fait passer de $\mathbb{R}^{v(i-1)}$ à $\mathbb{R}^{v(i)}$ et sera plongé dans l'ensemble $\mathcal{V}(T,v(0) \ldots v(T))$ des variables aléatoires sur $\mathbb{R}^{v(0)} x \ldots x\mathbb{R}^{v(T)}$, normé par une norme vectorielle N séparable de type $(N(x) = (N_t(x_t))_{t = 1 \ldots T})$ où les N_t sont des normes usuelles et où x_t est la projection de x sur $\mathbb{R}^{v(t)}$.

Remarque :

- Le cas déterministe, où f_t est un opérateur canonique (couramment l'addition, la soustraction, la multiplication, la division ou leurs représentations machine) ou une combinaison logique d'opérateurs canoniques, est le cas usuel pour un calcul effectué sur ordinateur.

Exemple :

y: = x*x -z peut être écrit :

x_1:=(x, z, x) = (a, b, c);
x_2:=(a*c, b) = (e, f);
x_3:= e-f ;

- L'introduction de l'aléa est pour le moment un artifice purement mathématique mais sera utile par la suite (lorsqu'on donnera une forme effective à notre théorie).
- Dans ce modèle, on plonge les opérateurs machine dans l'ensemble des opérateurs réels. On reprend là le parti de nombreux auteurs. Cela implique évidemment du point de vue expérimental d'observer les algorithmes "d'assez loin".
- Cette modélisation est très similaire à la représentation arborescente de Bauer [Bau].

B-2 Stabilité arithmétique d'un algorithme.

Il nous faut maintenant aborder l'étude de l'effet de la discrétisation machine sur un algorithme. Comme pour une discrétisation de type numérique, cette approximation est une perturbation de l'algorithme initial qui répond à la définition suivante.

Définition 1. Une perturbation p d'un algorithme A dans $\mathfrak{A}(T, v(0) \dots v(T))$ est une application p d'un espace vectoriel normé (E, N) dans $\mathfrak{A}(T, v(0) \dots v(T))$ qui vérifie $p(0) = A$.

Perturbation arithmétique.

Le fait que la discrétisation machine réponde à la définition précédente est immédiat. Il suffit par exemple de prendre pour $p(\varepsilon)$ l'exécution de A sur une machine travaillant sur $1-\log_2\varepsilon$ bits. En arithmétique virgule flottante, la perturbation "exécution machine" est une réalisation (un cas particulier) de la perturbation p^a définie par :

$$p^a : \mathbb{R}^{v(0)} * \dots * \mathbb{R}^{v(T)} \rightarrow \mathfrak{V}(T, v(0), \dots, v(T))$$
$$(e_t)_{t=0 \dots T} \rightarrow (X_t)_{t=0 \dots T}, \text{ avec } X_{t+1} = f_t(X_t) + \mathrm{Exp}(f_t(X_t)).e_{t+1} \text{ et } X_0 = x_0 + \mathrm{Exp}(x_0).e_0$$

où $\mathrm{Exp}(X)_i$ est égal à $2^{\exp(X_i)}$ et où "." désigne le produit tensoriel.

Voir l'exécution machine sous cette forme nous parait très instructif. Une telle définition redonne d'abord une certaine symétrie entre l'algorithme machine et l'algorithme théorique puisque l'algorithme "théorique" peut aussi s'écrire (sauf au voisinage des changements d'exposants) comme une perturbation de cette forme de l'algorithme machine. Cette propriété semble particulièrement intéressante dans une optique pratique puisque, lorsqu'on travaille avec une précision fixée, on a seulement accès à l'algorithme machine. Cette vision permet également de lier la stabilité par rapport aux arrondis d'un algorithme au problème sous-jacent (voir la suite du chapitre). On rappelle d'ailleurs que c'est cette structure de perturbation qui est adoptée dans certains modèles de contrôle d'erreur comme celui de Chesneaux. Nous appelerons cette perturbation la ***perturbation arithmétique.***

La qualification d'un algorithme consiste en l'étude de sa résistance par rapport à la perturbation "exécution machine", cas particulier de perturbation arithmétique. Cette résistance est communément appelée *stabilité.*

Comme on l'a vu dans le premier chapitre, on peut s'intéresser à la persistance (sous perturbation) de la *structure* mathématique des résultats fournis par l'algorithme, on parlera alors de stabilité qualitative [Cha1] ou structurelle [Tho], ou à la persistance de *l'ordre de grandeur* des résultats appelée stabilité quantitative. La plupart des utilisateurs se contentent (peut-être à tort) de l'étude de la stabilité quantitative. C'est donc ici ce sens que recouvrira sans précision le terme de stabilité. La stabilité par rapport à la perturbation p^a sera appelée stabilité arithmétique. Les définitions usuelles [Cha1] de stabilité utilisées sont au nombre de trois.

Notation : On notera par la suite X_T^e le résultat de l'exécution de p(e).

- **Définition 1:** A est stable pour p si l'application $e \rightarrow X_T^e$ est bornée sur un voisinage de 0.

Cette notion, utile par exemple pour des perturbations de processus itératifs n'est pas du tout adaptée pour l'étude de la stabilité arithmétique d'algorithmes finis qui sont toujours arithmétiquement stables en ce sens. On introduit donc une autre définition.

- **Définition 2 :** A est bien posé pour p si l'application $e \rightarrow X_T^e$ est continue en 0.

Cette seconde définition permet d'assurer la calculabilité de X_T sous perturbation. Cependant, elle est purement qualitative et ne débouche pas sur une mesure de la stabilité satisfaisante. C'est pourquoi nous donnons une troisième définition de stabilité utilisée dans la pratique [Laur][Mil].

- **Définition 3 :** A est lisse en 0 pour p si l'application $e \rightarrow X_T^e$ est C^1 au voisinage de 0.

Au contraire des algorithmes bien posés, dans le cas non dégénéré (différentielle $D_e(X_T^e)$ de X_T^e en 0 non nulle), la stabilité d'algorithmes lisses peut être mesurée. Dans ce cas, on a en effet pour e dans un voisinage V de 0 :

$$N_T(X_T^e - X_T) \leq n(e) * K_1^V \text{ où } K_1^V = \sup_{e' \in V} N_T(D_{e'}(X_T^e)[e'])$$

K_1^V est appelé conditionnement (absolu) de A dans V pour (N_T, n) et mesure la perte de précision due à une erreur unitaire. Si V est de diamètre assez petit le conditionnement peut être bien approché par la norme de la différentielle de X_T^e en 0 (c'est le conditionnement de Rice [Ric]).

Cette mesure est certainement la plus utilisée dans la littérature [Cha2], [Ric], [Dem], [Fle],[Mil]. Pour l'étude de la stabilité d'un *problème* ($p(e) = p^a(e/Exp(x_0),0,...,0)$, cf paragraphe II-3), on a aussi la notion de conditionnement relatif pour (N,n). C'est le conditionnement absolu correspondant aux normes (N_T', n') définies par :

$$N_T'(X)=\frac{N_T(X)}{N_T(X_T)} \text{ et } n'(e)=\frac{n(e)}{n(X_0)}$$

Exemple.

Considérons une matrice singulière A, et la résolution en précision infinie par l'algorithme de Gauss du système AX = b. Considérons alors la perturbation de cet algorithme consistant à l'appliquer pour résoudre le système $(A+E)X_e = b$ (perturbation des données). L'algorithme est lisse par rapport à cette perturbation et on a :

$$D_e (X_e) (0) = -A^{-1} . (A^{-1}b).$$

Par conséquent le conditionnement relatif (de Rice) est majoré par :

$$Cond(A) = N(A). N(A^{-1})$$

Comme on le sait, cond(A) est appelée conditionnement de la matrice A [Gol]. C'est la première notion de conditionnement apparue. Elle s'est imposée dès les débuts de l'informatique pour analyser l'effet sur le résultat d'une erreur sur les coefficients lors de la résolution d'un système linéaire. Rice a ensuite introduit [Ric] une notion générale de conditionnement pour les problèmes lisses qui, nous venons de le voir, s'adapte à toute perturbation lisse d'un algorithme. Nous verrons plus tard que F Chatelin [Cha1] a étendu cette mesure à certains problèmes bien posés. Nous la généraliserons à une plus grande classe d'algorithmes (algorithmes q-stables).

B-3 Stabilité du problème sous-jacent.

A un algorithme A est associé le problème sous-jacent de calcul de la fonction A(.). On peut aussi définir la stabilité de ce problème comme sa résistance à une perturbation de ses données, ce que nous précisons ci dessous :

Définition4 : Pour $p(e) = p^a(e, 0, \dots, 0)$ la stabilité de l'algorithme A par rapport à p sera appelée stabilité du problème sous-jacent à A.

Intuitivement cette stabilité conditionne celle de A par rapport aux arrondis. Cette liaison n'est cependant pas très facile à mettre en évidence. Le théorème suivant montre par contre bien le lien entre stabilité arithmétique d'un algorithme et stabilité du problème qu'il résout.

Théorème.

Si A est arithmétiquement lisse (respectivement stable ou bien posé) alors son problème sous-jacent est lisse (respectivement stable ou bien posé).

Démonstration.

p(e) est égal à $p^a(e, 0, \ldots, 0)$. Or si p^a est différentiable sur V alors elle est différentiable au sens de Gateaux (chacune de ses applications partielles est différentiable) p est donc *a fortiori* différentiable sur un voisinage de 0, V' (projection de V sur l'espace de perturbation des données).

De la même manière, si p^a est continue (respectivement bornée) alors p est continue (respectivement bornée).

C - MESURES DE LA STABILITE.

La mesure usuelle de la stabilité (conditionnement de Rice) n'est valable que pour des algorithmes ou des problèmes différentiables. Il faut en particulier que la solution perturbée converge vers la solution exacte à la même vitesse (en nombre de chiffres gagnés) que l'erreur machine élémentaire. Nous verrons ultérieurement que cela n'est pas vrai pour de nombreux algorithmes rencontrés en pratique. On peut également voir que cela n'est pas vrai pour certains problèmes classiques.

Exemple.

Prenons $A(a, b) = (x_1, x_2)$ où x_1 et x_2 sont les racines de $x^2 + ax + b = 0$. Supposons $a^2 - 4b = 0$ et étudions la stabilité du problème, c'est à dire le comportement en fonction de $(\varepsilon_1, \varepsilon_2)$ au voisinage de 0 de $(x_1(\varepsilon_1, \varepsilon_2), x_2(\varepsilon_1, \varepsilon_2))$ où $x_1(\varepsilon_1, \varepsilon_2)$ et $x_2(\varepsilon_1, \varepsilon_2)$ sont solutions de l'équation $x^2+(a+\varepsilon_1)x+(b+\varepsilon_2) = 0$.

On a : $x_1(\varepsilon_1,\varepsilon_2)= \dfrac{-a-\varepsilon_1 +\sqrt{\varepsilon_1^2+2a\varepsilon_1-4\varepsilon_2}}{2}$ et $x_2(\varepsilon_1,\varepsilon_2)= \dfrac{- a-\varepsilon_1 -\sqrt{\varepsilon_1^2+2a\varepsilon_1-4\varepsilon_2}}{2}$

Le problème n'est donc pas C^1.

Par ailleurs, la notion d'algorithme bien posé n'est pas susceptible ici de fournir une mesure de stabilité. Cela nous amène donc à chercher une définition intermédiaire. Reprenons le cas précédent. Pour les racines réelles, on peut par contre écrire (en négligeant le terme d'ordre 2) :

$$\left| x_i(\varepsilon_1,\varepsilon_2)+\frac{a}{2} \right| \leq \frac{\max_{j=1..2}|\varepsilon_j|+\sqrt{(2|a|+4)}\max_{j=1..2}\sqrt{|\varepsilon_j|}}{2}$$

ou encore au premier ordre en $\max_{j=1..2}\sqrt{|\varepsilon_j|}$:

$$\left| x_i(\varepsilon_1,\varepsilon_2)+\frac{a}{2} \right| \leq \frac{\sqrt{(2|a|+4)}\ \sqrt{\max_{j=1..2}|\varepsilon_j|}}{2}$$

$\sqrt{2|a|+4}$ apparait alors comme une mesure de stabilité qu'on peut encore appeler "conditionnement". Plus précisément c'est le plus petit nombre C tel que au premier ordre en $\max_{j=1..2}\sqrt{|\varepsilon_j|}$:

$$\left| x_i(\varepsilon_1,\varepsilon_2)+\frac{a}{2} \right| \leq \frac{C\sqrt{\max_{j=1..2}|\varepsilon_j|}}{2}$$

Les travaux de Chatelin sur la perturbation des problèmes de valeurs propres [Cha1] (problèmes similaires au problème cité en exemple) l'ont amenée ainsi à définir un conditionnement plus général pour ce type de problème où la solution n'est plus de l'ordre de la perturbation machine mais de l'ordre de sa racine pième. C'est de cette définition que l'on est parti pour donner une définition de stabilité plus générale.

C-1 q-stabilité.

Définition 5:

On dira que A est q-stable sur E_0 pour p si et seulement si :

$$\exists\, c > 0,\ \forall\, e \in E_0,\ N_T\,(X_T^e - X_T) \leq c.n(e)^q.$$

De la même manière, on dira que A est q-stable pour p si il est q-stable sur un voisinage de 0.

Remarque : F.Chatelin [Cha1] a introduit une q-stabilité particulière (Cas $q = \frac{1}{n}$ où n est un entier) pour étudier la stabilité de problèmes spectraux.

Propriété : Si q est strictement positif la q-stabilité de A pour p dans un voisinage de 0 implique que A est bien posé.

Cette nouvelle notion de stabilité étant définie, il semble alors intéressant de pouvoir définir une mesure de la stabilité dans l'ensemble des algorithmes q-stables. C'est pourquoi l'on introduit les notions de conditionnement ci dessous :

Définition 6:

Pour tout réel q, $C(E_0,q) = \sup_{e \in E_0}\left(\frac{N_T(X_T^e - X_T)}{[n(e)]^q}\right)$ est appelé *conditionnement d'ordre q* de A relatif à p sur E_0. $C(q) = \lim_{r \to 0} C(B(0, r), q)$ est alors appelé conditionnement d'ordre q de A pour p.

Rice s'est intéressé uniquement, dans le cas d'algorithmes lisses, à C(1) [Ric]. Cependant, cette notion est très générale puisqu'elle a un sens pour n'importe quel ordre et n'importe quel support. Par ailleurs on peut directement caractériser la q-stabilité grâce aux conditionnements. En effet, A est q-stable sur E_0 si et seulement si $C(E_0,q) \neq +\infty$ et de la même manière, A est q-stable si et seulement si $C(q) \neq +\infty$.

Exemple : q-diff stabilité.

Si $E = \mathbb{R}$ et si X_T^e est une fonction C^1 de la variable e^q dans un voisinage de 0, à dérivée non nulle en 0, alors A est q-stable dans un voisinage de 0 et la dérivée de X_T^e en 0 par rapport à e^q donne une bonne approximation du conditionnement de A d'ordre q dans un voisinage de 0. Dans ce cas, on dira que A est q-diff-stable. Un exemple de q-diff stabilité est donné par une perturbation polynomiale d'une équation polynomiale $Q(X) = 0$ de la forme $R(X, e) = 0$ [Duv].

Cas particulier de la stabilité arithmétique.

Dans le cas de la perturbation arithmétique, par définition même, la q-stabilité arithmétique dans un voisinage de 0 entraîne de plus la q-stabilité du problème. La démonstration est absolument identique à celle du théorème du paragraphe précédent.

C-2 Régularité.

Sous l'hypothèse d'une q-stabilité, il est alors intéressant de pouvoir calculer le plus grand q possible, c'est à dire celui qui donne la majoration la plus fine. Ce réel q_0 procure en effet une indication de la rapidité de convergence de la solution perturbée vers la solution exacte. On peut aussi voir cette mesure comme la mise en évidence d'un type de morphologie auquel notre algorithme est associé. q_0 est alors en un certain sens une mesure de la régularité de cette morphologie. C'est pourquoi l'on propose la définition suivante :

Définition 7: Sup {q tel que A q-stable pour p (resp sur E_0)} est appelé régularité de A par rapport à la perturbation p (resp sur E_0).

Remarques.

- Si r est la régularité de A, alors en augmentant la précision des calculs de 1 bit on obtient au moins r chiffres significatifs supplémentaires sur le résultat.
- Cette notion est très similaire à la notion d'ordre de convergence pour un schéma de discrétisation ou une itération.
- Si q_0 est la régularité de A, on remarque que pour q strictement inférieur à q_0 C(q) est nul, tandis que pour q strictement supérieur à q_0, C(q) est infini. La régularité de A peut donc être définie par sup $\{q \mid C(q) = 0\} = \inf \{q \mid C(q) = +\infty\}$.
- Le principe de la définition de la régularité d'un algorithme via la suite des conditionnements est tout à fait comparable au principe de la définition de la dimension de Hausdorff [Laus]. Tout comme tout ensemble a une mesure de Hausdorff, n'importe quel algorithme a une régularité.
- Si A est q-diff stable, q est la régularité de A.

Cas de la stabilité arithmétique.

Par construction, si A a pour régularité arithmétique q alors le problème sous-jacent admet une régularité $q_p \geq q$.

Ainsi, pour des algorithmes q-stables le couple régularité-conditionnement constitue une mesure de stabilité d'un algorithme par rapport à une perturbation.

C-3 Analyse inverse.

Comme nous l'avons déjà évoqué, nous aimerions aussi être capable de relier la stabilité d'un algorithme à celle du problème sous-jacent. L'outil classique d'analyse inverse (on dit aussi régressive) de Winkilson [Win] va être le support de ce paragraphe. Le principe de cette approche est de chercher à écrire la solution perturbée comme solution du problème exact pour des données perturbées. C'est ce que précise la définition suivante

• Définition 8 : On dira qu'on peut effectuer une analyse régressive de la perturbation p si pour $e \in E_0$, p(e) est égal à p^a (p'(e)/Exp(x_0),0,...,0), où p' est une perturbation des données. C'est à dire si le résultat de l'algorithme perturbé par p est le résultat de l'algorithme exact pour la perturbation p' des données.

Exemples.

• Pour un algorithme de résolution de $F(X) = x_0$, une analyse inverse est toujours possible. Il suffit de considérer $p'(e) = F\ (p(e)(x_0))-x_0$ [Cha1].
• Winkilson a montré la possibilité d'analyse inverse de l'algorithme QR (pour la perturbation arithmétique) en précision infinie [Win].

Lorsque l'analyse inverse est possible, on peut alors étudier la perturbation p'. On parlera alors d'étude de stabilité inverse et on aura les définitions suivantes.

Définition 4 :

• A est inversement bien posé pour p si et seulement si p' est continue en 0.
• A est q-inversement stable pour p si et seulement si :

$$\exists c, \forall e \in E_0,\ N(p'(e)) \leq n(e)^q . c$$

Ces notions sont intéressantes parce qu'elles conduisent au problème mathématique sous-jacent. (Voir [Cha1], [Cha2] et def 4). On a en effet le théorème qui suit.

Théorème.

Si un problème est q_p stable et si un algorithme de résolution A de celui ci est q_r-inversement stable alors A est $q_p.\ q_r$ stable. Pour un algorithme A q_r inversement diff-stable et un problème q_p-diff- stable, $q_p.q_r$ est de plus égal à la régularité de p.

La démonstration est immédiate.

La régularité inverse donne donc une indication pessimiste de la perte de stabilité induite sur le problème par l'algorithme. On remarque également que dans le cas de la stabilité arithmétique on a nécessairement (grâce aux propriétés précédentes) $q_r \leq 1$.

Exemple de stabilité inverse.

Pour une matrice assez bien conditionnée, l'algorithme de Gauss en précision infinie est inversement 1-stable [Gol].

Ainsi, l'introduction des notions de q-stabilité, q-diff stabilité, régularité, en complément de la notion classique de conditionnement nous permet de disposer d'une boite à outils pour mesurer la stabilité d'une vaste classe d'algorithmes bien posés soumis à des perturbations et spécialement aux perturbations arithmétiques. Dans les chapitres suivants, nous verrons comment, sous certaines hypothèses, des estimations de ces mesures théoriques peuvent être effectivement calculées. Mais avant de conclure, nous particularisons l'étude au cas des algorithmes analytiques et reprenons succintement le cas de la résolution d'équations que Chatelin a déjà traité [Cha2].

D - CAS PARTICULIERS.

D-1 Algorithmes analytiques.

Un cas particulier important est celui des algorithmes pour lesquels la perturbation peut s'exprimer comme fonction analytique des perturbations élémentaires. Dans ce cas on dira qu'on a un algorithme *analytique*. C'est le modèle retenu par certains auteurs [Che] pour la perturbation arithmétique (ou la perturbation arithmétique inverse) d'un "algorithme papier". Cela exclut évidemment la plupart des algorithmes contenant des tests. Les algorithmes analytiques sont évidemment de classe C^1 c'est à dire 1-stables mais pour de tels algorithmes nous pouvons encore affiner l'étude.

En effet dans ce cas on peut écrire :

$$X_T^e - X_T = F(e_1, e_2, \ldots, e_{T-1}) + G(e).$$

où F est une fonction linéaire de e et G une fonction de e contenant uniquement des termes d'ordre supérieur à 2.
Si on suppose les e_i aléatoires d'ordre infini (cas de Chesneaux [Che] par exemple), on peut donc écrire :

$$E(X_T^e - X_T) = F(E(e_0), E(e_1), \ldots, E(e_{T-1})) + G(e).$$

Où E(X) désigne l'espérance de la variable aléatoire X. Le terme F mesure donc l'effet sur le biais local d'une accumulation de biais d'arrondis locaux. Une forte instabilité accumulative est par exemple associée à des calculs de sommes. Dans le second terme, nous avons par contre le biais dû à un indéterminisme sur les erreurs locales c'est à dire aux non linéarités.
Pour cette raison, le terme linéaire F sera appelée ***l'instabilité accumulative***, le second terme sera lui appelé ***instabilité de volatilité*** (en suivant la nomenclature adoptée en finance [Ren]). Il nous parait important de séparer ces deux instabilités en particulier pour détecter la présence de non-linéarités ou tester l'importance de l'effet accumulatif, par exemple en vue de validation de méthodes de contrôle d'erreur basées sur des développements à l'ordre un.

D-2 Résolution d'équations.

Nous reprenons succintement ce cas traité par Chatelin et auquel s'intéresse toute l'analyse numérique. Etant donnée une équation $F(X_0) = Y_0$, avec F continue localement inversible au voisinage de Y_0, résolue de manière approchée sur ordinateur, on essaie d'analyser la stabilité de la solution informatique X_e. La méthodologie classique pour les discrétisations d'équations différentielles est basée sur la modélisation du calcul de X_e sous la forme $F_e(X_e) = Y_e$ (analyse régressive) où F_e est continue et localement inversible sur un voisinage de Y et sur la classique propriété **consistance + stabilité implique convergence"** qui peut être résumée par le théorème suivant :

Théorème.
Si (F_e^{-1}) est équicontinue (stabilité) et si quand e tend vers 0, F_e converge simplement vers F tandis que Y_e converge vers Y alors X_e converge vers X.

démonstration : Elle est classique [Sch] :

$$N(X_e-X)\leq N(F^{-1}{}_e(Y)-F^{-1}(Y))+N(F_e{}^{-1}(Y)-F_e{}^{-1}(Y_e))$$

Le premier terme converge vers 0 à cause de la consistance ainsi que le second terme à cause d'une part de l'équicontinuité et d'autre part de la consistance.

Si cette approche est complètement utilisable pour certaines discrétisations numériques pour lesquelles on a accès explicitement aux F_e et où on peut donc vérifier les conditions du théorème, elle ne peut par contre pas fonder totalement une étude de la stabilité par rapport aux discrétisations arithmétiques. Cependant, comme F. Chatelin l'a déjà montré [Cha2], elle peut sous certaines hypothèses servir de base à une mesure de stabilité partielle du calcul. Chatelin [Cha2] suppose par exemple les $(F_e{}^{-1})$ fonctions équilipschitziennes de $N_0(Y-Y_0)^{1/n}$. On a alors :

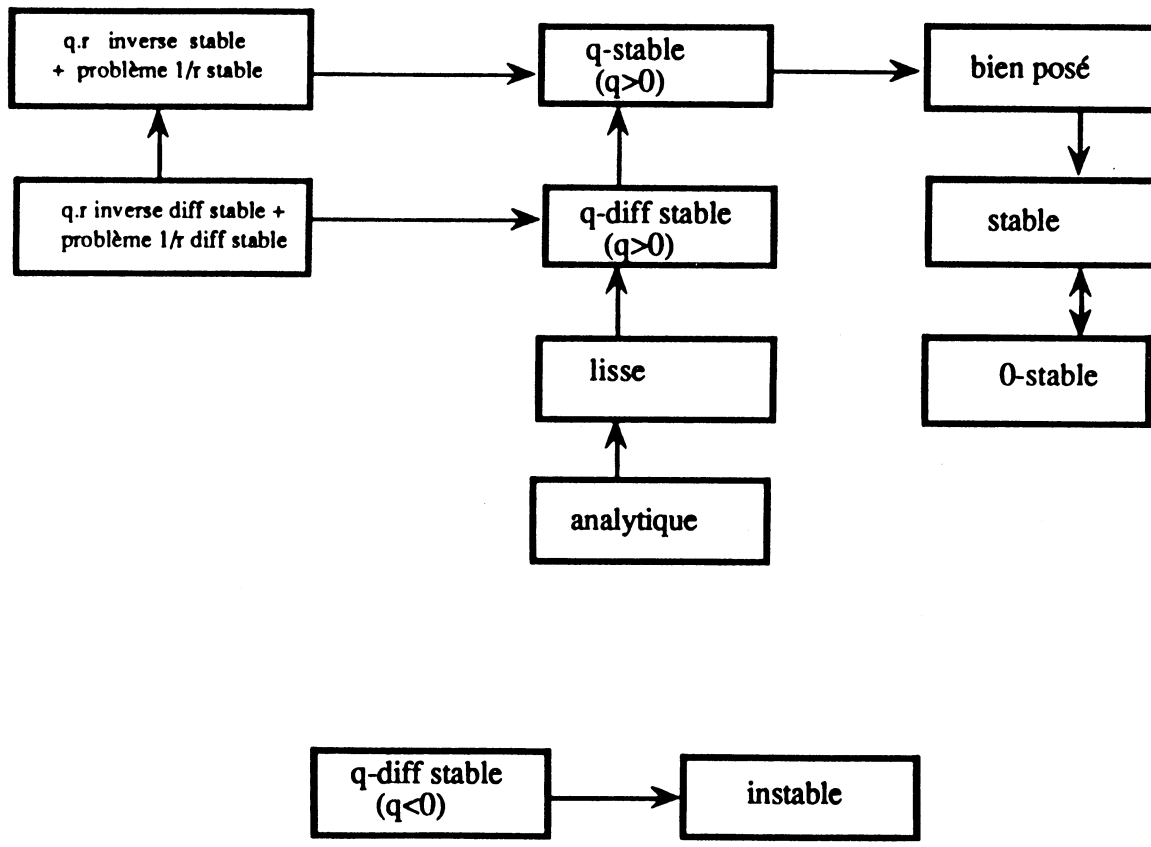
$$N(X_e-X)\leq N(F_e(Y)-F^{-1}(Y))+C.N_0(Y-Y_e)1/n$$

C, conditionnement du problème machine constitue donc une mesure de stabilité partielle.

E - CONCLUSION.

Nous venons donc d'introduire des outils mathématiques de mesure de la stabilité quantitative. Nous résumons dans le diagramme suivant, similaire à celui F Chatelin dans [Cha1] mais avec nos notions en plus, le lien entre les différents concepts de stabilité exposés :

Différentes stabilités ?



Parmi ces différents concepts, les notions de q-stabilité et de q-diff-stabilité retiennent particulièrement notre attention. Outre le fait qu'elles généralisent le cas modèle usuel des algorithmes réguliers, elles permettent de définir une mesure de la stabilité pour une large classe d'algorithmes et de problèmes. Elles serviront d'ailleurs de base à notre méthodologie "SCALP" de mesure expérimentale de la qualité arithmétique des logiciels. Il resterait maintenant à aborder le cas des problèmes mal posés pour lesquels une certaine stabilité qualitative peut exister.

BIBLIOGRAPHIE

[Bau] J BAUER, Computational graph and rounding error. SIAM J num anal, VOL 11 N°1, 1974.

[Cha1] F CHATELIN, Analyse statistique de la qualité numérique et arithmétique de la résolution approchée d'équations par calcul sur ordinateur. Etude F. 133, centre scientifique IBM FRANCE, avril 1988.

[Cha2] F CHATELIN, Résolution approchée d'équations sur ordinateur. Notes de cours de DEA de statistique, Université PARIS 6, 1989-1990.

[Che] J M CHESNEAUX, Etude théorique et implantation en ADA de la méthode CESTAC. Thèse de l'université PARIS 6, avril 1988.

[Dem] J W DEMMEL, On condition numbers and the distance to the nearest-ill posed problem number. Numer Math, 51, p251-289, 1987.

[Duv] D DUVAL, Développement de puiseux. Notes de cours de troisième année Ensimag, 1987.

[Fle] R FLETCHER, Expected conditioning, IMA J of Num Analysis, pp 247-273, 5, 1985.

[Fra] P FRANCOIS. Mesure de l'erreur arithmétique et perturbations des calculs. Rapport de recherche IMAG-TIM3, Janvier 1989.

[Gol] G GOLUB et C VAN LOAN. Matrix computation. John Hopkins Univ Press, Baltimore, 1983.

[Laur] P J LAURENT, Approximation et optimisation. Editions Hermann, Collection Enseignement des siences, PARIS, 1980.

[Laus] C LAUSBERG, Calcul numérique de la dimension fractale d'un attracteur étrange. Thèse de Doctorat de L'INPG, Grenoble 1987.

[Mil] W MILLER, Computer search for numerical instability. J of ACM, vol 22, N°4, pp 512-521, Octobre 1976.

[Ren] T RENAULT, Econométrie de la finance, Notes de cours de troisième année de l'ENSAE. Malakoff, 1988-1989.

[Ric] J RICE, A theory of condition. SIAM J Numer. Anal. 3, pp 287-310, 1966.

[Sch] L SCHWARZ, Topologie générale et analyse fonctionnelle. Editions Hermann, Collection Enseignement des siences, PARIS, 1980.

[Tho] R THOM, Modèles mathématiques de la morphogénèse. Editions Christian Bourgeois, Paris, 1981.

[Win] JH WINKILSON, The algebraic eigenvalue problem. Oxford University Press, 1965.

CHAPITRE IV

UN OUTIL DE QUALIFICATION DU LOGICIEL NUMERIQUE (noyau logiciel de la méthode scalp)

A - INTRODUCTION.

Comme nous l'avons déjà souligné dans notre introduction, les gros utilisateurs de calcul scientifique comme le CNES ressentent désormais le besoin de disposer d'un atelier de qualification de leurs logiciels numériques. Le but d'un tel outil semble tant d'aider à la conception des programmes qu'à la validation finale des produits.

Le point de vue probablement le plus répandu, et sans doute le plus pragmatique, consiste comme pour tout appareil fournissant des mesures, à chercher à instrumenter les logiciels de manière à ce qu'à tout résultat produit puisse être automatiquement associé une erreur sur celui ci. C'est l'objectif d'une méthode comme CESTAC [PV], [Che] et, dans une moindre mesure, du logiciel ACRITH [KM]. Cependant, lors des phases de conception, une analyse plus "mathématique" de la stabilité des logiciels semble être intéressante.

Le chapitre précédent nous a donné des bases théoriques pour une mesure de la stabilité d'un algorithme et du problème sous-jacent. Si on savait calculer analytiquement l'effet de perturbations de l'arithmétique ou des données sur un algorithme à exécuter, tous les indices de qualité définis dans ce chapitre seraient accessibles. Malheureusement, l'utilisation d'outils mathématiques de type analyse régressive-théories du conditionnement [Gen] se limite à des cas simples. Une approche expérimentale de l'analyse de la stabilité s'impose donc.

Dans cette optique, l'idée des méthodes de perturbation de procéder par sondage nous a paru intéressante, tout comme aux concepteurs de PRECISE (Precision Estimation and Control In Sientific and Engineering Computing) [Cha]. Il restait à définir les modalités pratiques d'implantation d'un système de sondage de la stabilité utilisable sur n'importe quel programme. En particulier la notion de perturbation suffisante. C'est l'objectif de ce chapitre. A partir d'idées empiriques, on définit ici les primitives logicielles à la base du système de sondage de la qualité numérique qui a été implanté au laboratoire TIM3 sur SUN3, en grande partie dès le début 1989, et qui a constitué le point principal de notre collaboration avec le CNES [DG] [Fra].

B - SONDAGE DE LA STABILITE.

B-1 Quels buts pour la qualification du logiciel.

Un logiciel numérique étant donné, il semble d'abord intéressant d'étudier sa résistance à une diminution de précision, c'est à dire à une perturbation de l'arithmétique (d'après le chapitre précédent, mesurable par la régularité et le conditionnement arithmétique du logiciel, véritables indices de qualité du programme). Mais comme nous l'avons déjà vu auparavant, la qualité d'un algorithme est relative à la difficulté du problème qu'il résout. On ne pourra pas par exemple, trouver d'algorithme précis pour calculer une fonction discontinue ou mal conditionnée. Il semble donc également important de relier la stabilité du logiciel étudié à celle du problème mathématique sous-jacent (d'après le chapitre précédent, la régularité et le conditionnement du problème). C'est pourquoi, l'on doit pouvoir étudier l'effet de perturbations des données sur les résultats. Ces objectifs sont ceux de l'outil logiciel dont le canevas est décrit dans de ce chapitre.

Parallèlement, le repérage des zones arithmétiquement critiques dans un programme et la recherche d'une borne d'erreur resteront toujours à l'ordre du jour. Mais comment aborder ces questions?

B-2 Un système arithmétique sondant.

De tels objectifs ne semblent accessibles que dans la mesure où l'on est capable d'observer séparément des perturbations de l'algorithme (au niveau de chaque opération et pour différentes ampleurs de perturbations) et du problème. La plupart des systèmes de calcul ne semblent pas l'autoriser. L'arithmétique virgule flottante, figée et déterministe, ne donne aucune information sur l'effet de ses propres perturbations. On peut tout au plus jouer sur les quelques précisions disponibles, ce qui est peu compte tenu des contraintes matérielles. L'arithmétique d'intervalles ne permet pas d'observer séparément des perturbations sur des données.

On peut par contre songer à étudier la stabilité (y compris éventuellement pour l'algorithme théorique) à l'aide d'un système de calcul multiprécision. Cependant, la portée du calcul multiprécision est limitée par des problèmes matériels.

Qu'en est t-il des possibilités offertes par l'utilisation de perturbations des calculs? Rappelons d'abord que l'idée qui sous tend l'utilisation d'une arithmétique perturbée est de tester empiriquement la stabilité. Mais peut-on accéder facilement en machine à des perturbations arithmétiques de différentes ampleurs?

Nous remarquerons d'abord qu'on effectue une perturbation arithmétique à chaque fois qu'on réalise une perturbation au dernier bit du type de celles définies au deuxième chapitre :

$$(I)\quad P_a(x) = x + \alpha\,(t,q).\ 2^{exp(x)-L}.$$

où prob $(\alpha(t,q) = 1) = t$, prob $(\alpha(t,q) = -1) = q$, prob $(\alpha(t,q) = 0) = 1 - t - q$

Or une telle arithmétique est facilement implantable à partir d'une arithmétique standard puisqu'il suffit d'effectuer les opérations normalement, en modifiant ensuite le dernier bit du résultat. Enfin, ce lot d'arithmétiques procure bien un système de perturbations d'ampleurs variables de l'algorithme machine. En effet, en se restreignant à des arithmétiques stochastiques symétriques (t = q), le paramètre t est une mesure de l'ampleur de la perturbation de l'algorithme initial. Ainsi, la généralisation de la perturbation de la Porte et Vignes [PV] à des amplitudes quelconques permet de disposer d'un système de perturbations arithmétiques d'ampleurs variables de n'importe quel logiciel numérique et fournit ainsi un système arithmétique sondeur de stabilité.

De la même manière, on peut envisager d'implanter une perturbation de type :

$$(II)\quad p_m(x,q) = A_q(x)\ \text{(arithmétique à précision paramétrée)}$$

où A_q est un arrondi sur q bits ($q < n$).

Remarque : On pourrait enfin envisager d'implanter la perturbation aléatoire

$$p_q(x) = x + x.\left(U\left[-2^{-L-1+exp(x)+q},\ 2^{-L+q+exp(x)}\right]\right)\ \text{(arithmétique de Hull et Swenson [HS])}$$

où L est le nombre de chiffres de mantisse, exp(x) l'exposant de x et où U[a,b] désigne un échantillon d'une variable aléatoire uniforme sur [a,b].

Trois systèmes arithmétiques simples permettent donc d'accéder à l'effet de perturbations arithmétiques sur l'algorithme machine. Le premier semble *a priori* plus intéressant dans la mesure où l'ordre de grandeur des perturbations est égal à celui des erreurs d'arrondi effectives. Il permettra aussi de faire des hypothèses probabilistes sur une solution obtenue avec un arrondi aléatoire ayant la précision de la machine et donc de valider des modèles paramétriques. Mais bien que l'aspect aléatoire des perturbations offre une certaine sécurité (à cause du terme de variance qu'il induit dans l'erreur), les erreurs obtenues seront sans doute sensibles à la qualité du générateur aléatoire. Au contraire les deux autres arithmétiques permettront probablement de visualiser un comportement de stabilité moyen. Quoiqu'il en soit, un tel lot d'arithmétiques parait être un outil intéressant pour l'étude de la stabilité arithmétique des logiciels numériques. On note de plus qu'en appliquant de telles perturbations aux seules données du problème et non plus aux calculs on dispose de la même manière d'un outil de mesure de la stabilité des problèmes. On retient donc ces arithmétiques perturbées comme base d'un système de contrôle de la stabilité. Une méthodologie détaillée d'utilisation de ces perturbations doit bien sûr être encore explicitée. Cela sera l'objet du chapitre suivant.

Cependant, on peut déjà donner l'optique d'utilisation d'un tel système arithmétique :

- *Pour visualiser la stabilité de l'algorithme :*

Pour une structure de perturbation p choisie
Pour différentes ampleurs
Si la perturbation est stochastique
pour différents échantillons
*Exécuter A en remplaçant chaque opérateur élémentaire **
*par l'opérateur *p perturbé.*
Evaluer l'erreur par rapport au calcul non perturbé sur chaque variable de sortie.

- *Pour évaluer la stabilité du problème :*
Effectuer les mêmes opérations en limitant la perturbation aux données.

Remarque :
Nous noterons bien qu'*a priori* une telle méthode ne sondera en premier ressort que la stabilité de l'algorithme machine et pas directement celle de l'algorithme mathématique dont il est issu. C'est là une différence avec l'utilisation d'arithmétiques perturbées faite par d'autres auteurs.

B-3 Conclusion.

On a donc maintenant défini la philosophie générale d'un système arithmétique paramétré d'étude de la stabilité. Il reste maintenant à expliciter les modalités pratiques d'implantation de ce système. C'est l'objectif du paragraphe suivant. Nous reviendrons ensuite en conclusion sur l'utilité de toutes les primitives développées.

C - UN OUTIL D'EVALUATION DE LA QUALITE NUMERIQUE.

C-1 Principales primitives de l'outil.

Un algorithme étant donné, il faut donc d'abord pouvoir réaliser les perturbations de cet algorithme définies dans le paragraphe précédent. Le premier module, noyau de l'outil, consiste donc en un lot d'arithmétiques perturbées et de procédures de perturbation des données. Les perturbations ainsi disponibles doivent de plus pouvoir être implantées de manière automatique sur n'importe quel programme sans que l'utilisateur ait dans l'écriture des programmes à faire référence à de nouvelles instructions à chaque fois qu'il appelle une opération arithmétique.

Un deuxième module (postcompilateur [DG]) permet donc d'implanter automatiquement des perturbations arithmétiques dans des séquences d'instructions choisies.

Nous avons vu que la plupart des méthodes d'analyse de l'erreur arithmétique basées sur une arithmétique perturbée supposaient un modèle probabiliste paramétrique pour la solution de l'algorithme perturbé. Il nous a donc semblé utile (ne serait ce qu'à des fins de test des autres méthodes) de se doter d'un outil de visualisation de densités d'échantillons, troisième module de l'outil.

Il nous a également fallu choisir une machine cible et un langage de programmation dans lequel les programmes à étudier seraient écris. Afin de pouvoir traiter des programmes de calcul du CNES, le langage retenu a été Fortran 77. Nous verrons cependant que l'outil peut très vite s'adapter à n'importe quel autre langage. La machine utilisée a été un SUN3.

C-2 Le module de perturbation.

Il consiste en 3 bibliothèques :

• Une bibliothèque contenant deux procédures de perturbations des données réalisées en Fortran 77 et s'appliquant respectivement à des données réelles en simple et en double précision. Ces procédures réalisent les perturbations :

$$p_d(x,q) = x + U\left[-2^{-L+q+\exp(x)}, 2^{-L+q+\exp(x)}\right]$$

où L est le nombre de chiffres de mantisse, q est le nombre de bits perturbés, exp(x) est l'exposant de x et où U[a,b] est un échantillon d'une variable uniforme sur [a,b] engendré à partir du générateur aléatoire par défaut de la machine et d'un entier d'initialisation de celui ci fourni par l'utilisateur.

• Une bibliothèque d'arithmétiques stochastiques au dernier bit (cf (I)). Cette bibliothèque contient des procédures d'addition, de multiplication, de division et de soustraction perturbées.

Cette bibliothèque a été réalisée en assembleur sur le coprocesseur arithmétique du SUN3 pour des raisons de rapidité. Les procédures d'arithmétique perturbée sont évidemment les mêmes pour toutes les probabilités de perturbation qui constituent simplement un paramètre.

• Une bibliothèque d'arithmétiques à précision paramétrée en arrondi au plus près sur q bits q<L (cf II). Cette bibliothèque contient des procédures d'addition, de multiplication, de division et de soustraction perturbées. Elle a également été réalisée sur le coprocesseur arithmétique du SUN3. Les procédures d'arithmétique perturbée sont évidemment là aussi les mêmes pour toutes les amplitudes de perturbation qui constituent simplement un paramètre.

C-3 Module d'histogramme.

Un échantillon X[1 ... N] de N réalisations d'une variable aléatoire (par exemple différentes réalisations de la solution pour une arithmétique stochastique au dernier bit) étant donné dans un fichier, ce module réalise le calcul de la moyenne et de la variance empirique de celui ci en "précision étendue" et l'estimation de la densité de cette variable aléatoire. Pour l'estimateur de la densité, nous avons choisi au départ une méthode adaptée à une densité régulière: la méthode du noyau à pas automatique [Ber] [DG]. Ce choix s'inscrivait dans le cadre d'une volonté de test "objectif " de modèles probabilistes de l'effet d'une arithmétique stochastique. Ces modèles présupposent souvent une loi normale pour le résultat de l'algorithme perturbé. Pour tester à l'oeil" cette normalité, il paraissait donc intéressant d'avoir un outil de visualisation de densité adapté à une loi normale. Par la suite, il a été inclus dans ce module la possibilité de choisir une estimation de la densité classique de type histogramme avec pas manuel afin de pouvoir également représenter les densités discrètes apparaissant dans la pratique (Voir chapitre II).

C-4 Implantation automatique.

Il nous fallait pouvoir remplacer dans une portion de programme choisie par l'utilisateur chaque affectation de type "Y:= *X", où * opère sur des vecteurs, par l'affectation "Y=*p(X)". C'est à cet effet qu'un postcompilateur a été écrit. Le principe de son fonctionnement est le suivant :

Premier principe (sur l'écriture des programmes) :
A chaque fois que l'utilisateur veut perturber une portion de programme avec une structure de perturbation p et une amplitude a, il ajoute une procédure d'initialisation I(a) au début de la portion de programme considérée. Toute la portion de code qui se trouvera avant l'appel de la procédure I suivant sera alors exécutée avec la perturbation d'amplitude a. Pour des raisons d'homogénéité, les portions de code à ne pas perturber devront être labelisées avec I(0).

Deuxième principe (sur la compilation) :
Il faut alors demander un mode de compilation spécifique. On génère d'abord le code assembleur. On remplace ensuite dans ce code toutes les fonctions arithmétiques qu'on a choisi de perturber par les fonctions d'arithmétique perturbée correspondantes. On génère enfin le code perturbé exécutable.

D - CONCLUSION.

Nous venons de définir dans ce chapitre les principes d'un outil logiciel d'arithmétique perturbée pour la mesure de la stabilité. On peut légitimement se demander si toutes les perturbations introduites sont vraiment nécessaires. En conclusion, nous donnons quelques éléments complémentaires de justification de nos choix.

D-1 Pourquoi perturber les opérateurs ?

Arguant du coût des perturbations des opérateurs arithmétiques, on peut d'abord se demander si un outil d'analyse de la stabilité du problème (c'est à dire des perturbations uniquement des données) ne serait pas suffisant pour visualiser la stabilité d'un logiciel. On rappelera que Larson a montré rigoureusement qu'une analyse inverse 1-stable des perturbations n'est pas toujours possible. Perturber seulement les données ne visualise *a priori* qu'un certain domaine de l'erreur finale. Plus pratiquement, même en supposant les perturbations des données suffisantes en théorie, l'amplitude nécessaire pour la visualisation d'une instabilité de l'algorithme peut être rendue aussi grande que voulue. Il suffit pour s'en convaincre de penser à un algorithme constitué de la composition d'un algorithme très contractant et d'un algorithme très instable par rapport aux données.

Exemple. $x_{n+1} = \frac{x_n + b}{2}$, $n = 0, \ldots, p$; $\quad x_{n+1} = 2(x_n - b)$, $\quad n = p, \ldots, q$.

Pour toute perturbation de la donnée x_0 et pour p assez grand, cette perturbation est invisible à l'étape p et donc sur le résultat final. Au contraire pour q assez grand l'erreur d'arrondi peut être aussi grande que voulue. On a par exemple pris $b = 0.5$, $x_0 = 3.14$, $p = 100$, $q = 300$. Les perturbations du seul x_0 sur 5 bits se sont avérées invisibles tandis que tous les chiffres du résultat se révélaient bien instables lorsqu'on utilisait la perturbation de Vignes.

Ces arguments justifient à nos yeux l'utilisation de perturbations des opérateurs.

D-2 Pourquoi ne pas considérer uniquement la perturbation de Vignes ?

Il ne s'agit pas pour nous de rejeter la méthode de Vignes. La mise en évidence grâce à des exemples ou des arguments théoriques tels ceux donnés dans les pages précédentes des limitations possibles de la méthode ne l'autorise nullement. Pour notre part nous avons plus cherché à utiliser un outil de perturbation des calculs comme un outil de révélation des instabilités arithmétiques potentielles d'un algorithme machine que comme un outil d'évaluation de l'erreur arithmétique à une précision donnée. C'est dans cette optique que nous avons été amenés à envisager différentes amplitudes de perturbation.

BIBLIOGRAPHIE

[Ber] BERLINET, Estimation fonctionnelle, Notes de cours de DEA de statistique. Ensimag, Grenoble, 1986-1987.

[Cha] F CHATELIN, Analyse statistique de la qualité numérique et arithmétique de la résolution approchée d'équations par calcul sur ordinateur. Etude F. 133, centre scientifique IBM FRANCE, avril 1988.

[Che] J M CHESNEAUX, Etude théorique et implentation en ADA de la méthode CESTAC. Thèse de l'université PARIS 6, avril 1988.

[DG] A DOUET et F GERARD, Analyse statistique d'un projet d'analyse de la qualité numérique, Rapport de projet d'ingénieur, ENSIMAG, 1989.

[Fra] P FRANCOIS, mesure de l'erreur arithmétique et perturbation des calculs. Rapport de recherche TIM3-IMAG N° 767-M, Juin 1988.

[Gen] W.M GENTLEMAN, Error analysis of QR decomposition by Givens transformations. Linear algebra and its applications 10, pages 189-197, 1975.

[HS] HULL et SWENSON, Test of probabilistic models for propagation of roundoff error. Communication of ACM Vol 9 N° 2, 1966.

[KM] U.W KULISCH et W.L MIRANKER, The arithmetic of the digital computer, a new approach. SIAM rev, vol 28, N° 1, mars 1986.

[PV] M LA PORTE et J VIGNES, Error analysis in computing. Information processing, 1974.

[Vig] J VIGNES, New methods for evaluating the validity of the results of mathematical computations. Mathematiccs and computers in simulation, Transaction of IMACS, Vol XX, N°4, pp 227-249, Décembre 1978.

CHAPITRE V

STABILITE ET CONDITIONNEMENT DES ALGORITHMES PAR PERTURBATION : SCALP

A - INTRODUCTION.

Nous avons déjà souligné que, dans l'optique du contrôle de la qualité arithmétique, il nous semblait intéressant de disposer d'autres indices de qualité qu'une borne d'erreur sur les résultats. Deux types d'informations paraissent en particulier être recherchées par les utilisateurs. D'une part la qualité du logiciel au regard de la difficulté du problème qu'il résout. D'autre part l'effet d'un changement de précision, sur les opérations arithmétiques ou sur les données, sur le résultat (par exemple dans l'optique du passage sur calculateur embarqué), que l'on peut mesurer par exemple par la perte d'information (en nombre de bits) sur le résultat par nombre de bits de calcul en moins.

La théorie de la stabilité arithmétique que nous avons formulée précédemment (cf chapitre 3) donne un sens autre qu'empirique à cette demande. La régularité arithmétique d'un algorithme (resp. d'un problème) est par exemple une mesure asymptotique de la perte d'information sur le résultat par perte d'information unitaire lors des opérations (respectivement par perte d'information unitaire sur les données). Par ailleurs la régularité du problème est, associée à son conditionnement, une bonne mesure de la difficulté de celui ci. Enfin, la différence entre la régularité du problème et celle de l'algorithme est un indicateur de la qualité de l'algorithme compte tenu de la difficulté du problème.

Malheureusement, l'obtention systématique de ces indices de qualité par une étude *a priori* est impossible à systématiser à cause de la complexité des phénomènes entrant en jeu. Nous avons affaire à des phénomènes discrets, mais les ensembles sur lesquels nous travaillons sont trop vastes pour appliquer des outils combinatoires. Il faut donc songer (comme pour le calcul d'une borne d'erreur) à l'estimation expérimentale de cette information.

Dans le chapitre précédent, nous avons défini un système logiciel d'arithmétiques perturbées. Dans ce chapitre nous définissons et illustrons une méthodologie "SCALP" d'utilisation de ce système pour mesurer effectivement la stabilité (telle qu'elle a été définie au chapitre 3) par inférence statistique.

Nous proposons d'abord des estimateurs de stabilité qui sous des hypothèses que nous explicitons sont des indicateurs du conditionnement et de la régularité d'un logiciel.

Nous illustrons ensuite ces estimateurs par quelques exemples en montrant en particulier comment ils permettent d'obtenir des résultats qualitatifs intéressants pour la comparaison d'algorithmes.

On ébauche enfin une heuristique de test de certaines hypothèses du modèle de Chesneaux [Che].

B - SCALP *(Stabilité et Conditionnement Arithmétique des Logiciels par Perturbation).*

B-1 Introduction.

Un algorithme machine produisant un résultat réel x étant donné, on aimerait, pour évaluer sa stabilité conformément au chapitre 3, donner des estimations de la régularité et du conditionnement de cet algorithme ainsi que de la régularité et du conditionnement du problème sous-jacent. Pour cela, nous savons qu'il est nécessaire d'effectuer des perturbations arithmétiques de l'algorithme et des données. C 'est à cet effet que nous avons introduit un logiciel de perturbation dans le chapitre précédent. Nous allons maintenant donner des hypothèses en quelque sorte "autovérifiables" sous lesquelles cet outil nous permet d'obtenir des indicateurs de conditionnement et de régularité.

B-2 Un outil statistique : la régression loglinéaire.

Considérons une fonction réelle f d'une variable réelle positive e telle que, au voisinage de 0, (f(e)-f(0)) soit équivalent à $C.e^q$. Les économètres semblent s'accorder à penser que si on dispose de plusieurs échantillons $f(e_i)_{i = 1..n}$ de f au voisinage de 0 [Maz], un modèle de régression linéaire du type :

$$(M)\ y_i = \log\ (f(e_i)-f(0)) = \log\ C + q.\log\ (e_i) + u_i\ = a + bx_i + u_i,$$
$$E\ (u_i) = 0,\ Var\ (u_i) = \sigma^2.$$

est un bon modèle des y_i. Nous ne rentrerons pas dans une justification de ce point de vue. Il semble d'ailleurs que se soit essentiellement un fait d'expérience. Nous prendrons donc par la suite cette hypothèse comme acquise.

Dans le cadre d'un tel modèle, on peut alors estimer q et log C par les formules :

$$\hat{q} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\widehat{\log C} = \bar{y} - \hat{q}.\bar{x}$$

où $\bar{y}$ et $\bar{x}$ désignent les moyennes empiriques de (x_i) et (y_i); c'est à dire

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

L'avantage expérimental d'un tel modèle est, outre sa simplicité de mise en oeuvre, de pouvoir en un certain sens s'autovalider. Même s'il faut prendre cette affirmation avec une certaine flexibilité, le coefficient de significance de la régression R^2, défini par :

$$R^2 = \frac{\left(\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})\right)^2}{\left(\sum_{i=1}^{n} (x_i - \bar{x})^2\right)\left(\sum_{i=1}^{n} (y_i - \bar{x})^2\right)}$$

doit en effet être proche de 1 dès que la régression est valide. Chaque fois qu'on aura un coefficient trop faible (mettons plus petit que 0,8 [Maz]), on concluera à l'inadéquation du modèle.

Considérons maintenant le cas d'une perturbation d'une seule variable réelle positive e d'un algorithme A, comme par exemple l'exécution de l'algorithme en arithmétique stochastique symétrique, en arrondi au plus près avec une précision paramétrée inférieure à la précision initiale ou sur des données perturbées.

Si la norme N (X(e) -X) de la différence entre un résultat réel perturbé et le résultat non perturbé est équivalente au voisinage de 0 à Ce^q, c'est à dire si :

$$N(X(e) - X) \approx Ce^q \text{ au voisinage de } 0$$

alors, d'après ce qui précède q et C sont donc estimables par régression log linéaire. Or dans ce cas, q représente, par définition (cf chapitre III), la régularité de l'algorithme par rapport à la perturbation considérée et C le conditionnement. L'outil régression linéaire fournit donc, sous de larges hypothèses, des estimations de la régularité d'un algorithme par rapport à des perturbations d'une variable réelle.

Cette approche s'adapte en particulier à des perturbations particulières associées aux perturbations définies au chapitre précédent. Cela nous permet de proposer maintenant des estimations des régularités et des conditionnements d'un algorithme par rapport à ces perturbations particulières simulables en machine.

B-3 Mesures de la stabilité.

Notations et conventions préliminaires, rappels :

Nous noterons L le nombre de chiffres de mantisse de la représentation virgule flottante utilisée. Par la suite nous supposerons, sans perte de généralité, qu'on travaille en base 2 et que les résultats auxquels on s'intéresse sont de dimension un.
Le résultat de l'algorithme perturbé pour une valeur e du paramètre de perturbation sera noté X(e). Le résultat exact sera noté X. Par ailleurs, pour pouvoir interpréter les erreurs observées sur le résultat en terme de nombre de chiffres significatifs, la norme utilisée sur l'espace des résultats sera la norme L_r^2 définie par :

$$|Y|_{L_r^2} = \frac{\sqrt{E\{Y^2\}}}{|X|}$$

On rappelle que dans le chapitre précédent, nous avons introduit un environnement logiciel ayant la possibilité de simuler les trois perturbations suivantes d'un algorithme :

1) Perturbation p_d : brouillage uniforme des données :

L'algorithme $p_d(2^{-L+m})$ où m est un entier choisi est obtenu à partir de l'algorithme initial en brouillant les données uniformément sur leurs m derniers bits, c'est à dire en remplaçant toute donnée x par la donnée perturbée :

$$p_d(x,q) = x + U\left[-2^{-L+m+exp(x)}, 2^{-L+m+exp(x)}\right]$$

où exp(x) est l'exposant de x dans sa représentation virgule flottante et où U[a,b] est un échantillon d'une variable aléatoire uniforme sur [a,b]. Ceci de manière indépendante sur des données indépendantes.

2) Perturbation p_S : Passage en arithmétique stochastique centrée :

L'algorithme $p_S(e)$ est obtenu à partir de l'algorithme initial en remplaçant tout opérateur élémentaire * par un opérateur perturbé au dernier bit $*_p$ défini par :

$$*_p(x) = p(*x)$$

où :

$$p(X) = X + \alpha(e).2^{-l+exp(X)}$$

où α est une variable aléatoire à valeurs dans 0, 1, -1 telle que :

$$prob(\alpha(e) = 1) = prob(\alpha(e) = -1) = \frac{e}{2}, \quad prob(\alpha(e) = 0) = 1 - e$$

3) Perturbation p_m : passage de l'algorithme en précision paramétrée :

L'algorithme $p(2^{-L+m+exp(x)})$ est obtenu à partir de l'algorithme initial en arrondissant tout résultat intermédiaire sur m bits de moins que dans le calcul non perturbé, ce qui revient à calculer sur m bits de moins.

Nous noterons q_s, q_d, q_m les régularités de l'algorithme étudié par rapport à ces perturbations, et noterons C_s, C_d, C_m leurs conditionnements.

Hypothèse (H) de notre analyse expérimentale.

Pour se placer dans les conditions d'application de l'analyse par régression du paragraphe précédent, nous supposerons que la norme de la différence entre le résultat perturbé et le résultat non perturbé vérifie :

$$N(X(e) - X) \approx Ce^q \text{ au voisinage de } 0.$$

Remarque: C'est en particulier le cas lorsque l'algorithme est q-diff stable par rapport à la perturbation considérée.

Pour la perturbation stochastique p_S, nous supposerons même le modèle de régression :

$$N(X(e)-X) = C_s.e^{q_s} + u_i \quad E(u_i = 0), \text{ var } u_i = \sigma^2$$

valable pour toute probabilité de perturbation.

D'après le paragraphe précédent, on peut alors se proposer d'estimer les régularités et les conditionnements de l'algorithme par rapport à une de ces perturbations en régressant les logarithmes des erreurs y_i (entre le résultat perturbé et le résultat non perturbé), obtenues pour différentes valeurs du paramètre de perturbation sur les logarithmes x_i du paramètre de perturbation associé.

Remarque importante.

Dans le cas des perturbations p_d ou p_m, les paramètres de perturbation e_i sont de la forme 2^{-L+m_i}. On peut donc écrire :

$$\log e_i = (-L + m_i) \log (2)$$

La régression des y_i sur les x_i équivaut alors à celle des y_i sur les m_i. Les coefficients de régression (logC,q) de la première régression se déduisant de ceux de la deuxième (a,b) par les formules :

$$a = -q.L.\log(2)+\log C$$
$$b = q.\log(2)$$

La régularité et le conditionnement de l'algorithme pour p_S ou pour p_d peuvent donc s'obtenir en régressant les logarithmes y_i des erreurs obtenues sur les m_i.

Cela nous amène sous l 'hypothèse (H) précédente à proposer les estimateurs de régularité et de conditionnement d'un algorithme suivants :

Estimation de q_d et C_d :

Pour différentes valeurs de m (m_i) "assez petites "

pour différents échantillons

perturber les mj derniers bits des données uniformément

évaluer l'erreur (relative quadratique moyenne) sur le résultat

(la référence étant le résultat non perturbé) Ej

Calculer les coefficients (a,b) de la régression de log(Ej) sur mj

$q_d = b/\log(2)$

$C_d = e^{a} * (2^{L})^{q_d}$

Estimation de q_s et C_s :

Pour différentes probabilités de perturbation (α_j)
pour différents échantillons
perturber l'algorithme par $p_s(\alpha_j)$
évaluer l'erreur quadratique moyenne sur le résultat
(la référence étant l'arrondi au plus près) E_j
Calculer les coefficients (a,b) de la régression de log(Ej) sur α_j
$q_s = b$
$C_s = e^a$

Estimation de $q_m$et C_m :

Pour différentes valeurs de nombres de bits d'arrondis en moins (m_j)
exécuter l'algorithme en arrondi sur m_j bits
évaluer l'erreur sur le résultat
(la référence étant l'arrondi au plus près) E_j
Effectuer une régression linéaire de(log(Ej))sur (m_j)
$q_m = b/log(2)$
$C_m = e^a * (2^L)^{q_m}$

Nous venons donc de proposer une méthode inspirée de l'économétrie pour estimer la régularité et le conditionnement d'un algorithme par rapport à des perturbations machine introduites au chapitre précédent. Cette estimation est *a priori* valable théoriquement pour une vaste classe d'algorithmes abstraits. En particulier pour les algorithmes q diff-stables. Reste à étudier la validité pratique d'un tel modèle et à préciser le lien entre les mesures de stabilité étudiées dans ce paragraphe et celles du chapitre 3.

B-4 Résultats fondant nos estimations de stabilité.

Nous donnons maintenant un certain nombre de résultats théoriques spécifiant le rapport entre les régularités et conditionnements arithmétiques d'un algorithme ou de son problème sous-jacent (pour des normes bien choisies) et les régularités q_s, q_p, q_m et conditionnements C_s, C_d,C_m dont nous avons proposés des estimateurs au paragraphe précédent.

Par la suite :

• ε_m désignera le "macheps" (*machine epsilon*) de la machine sur laquelle nous travaillons, c'est à dire le nombre $1_+ - 1$, où 1_+ est le successeur de 1 en machine.

• q_a désignera la régularité arithmétique de l'algorithme auquel on s'intéresse, $C_a(q)$ son conditionnement arithmétique d'ordre q, q_p et $C_p(q)$ la régularité et le conditionnement d'ordre q de son problème sous-jacent.

Pour les définitions générales de la régularité et du conditionnement ainsi que des perturbations arithmétiques et du problème, nous renvoyons le lecteur au chapitre 3. Nous rappelons simplement qu'on peut interpréter la régularité comme le nombre de bits maximum gagnés sur le résultat pour un bit de précision supplémentaire sur l'approximation de l'algorithme.

L'espace des résultats sera comme précédemment normé par la norme L^r_2, l'espace des paramètres de la perturbation arithmétique sera lui normé par la norme L_2 :

$$\left[\frac{\sum_{i=1}^{n} |e_i|^2}{n}\right]^{\frac{1}{2}}$$

où n est le nombre d'arrondis effectués au cours du calcul. Enfin, l'espace des paramètres de la perturbation du problème sera normé par la norme :

$$\sup_{i=1..n}\left(\frac{|e_i|}{2^{\exp(d_0^i)}}\right)_{i=1..N}$$

où d_0 désigne le vecteur des N données réelles du problème, et d_0^i sa $i^{ème}$ composante.

On rappelle d'abord qu'on a toujours : $q_a \leq q_p$. On démontre maintenant de nouveaux résultats de ce type qui nous permettrons d'estimer q_p et q_a en pratique.

Théorème sur la stabilité du problème.

$$q_p \leq q_d$$

Pour tout réel q, $\quad C_d(q) \leq \sqrt{\frac{N}{N+2q}}\, C_p(q)$

démonstration :

Dans cette démonstration A(d) désignera l'exécution de l'algorithme A pour la donnée d.
Considérons q tel que le problème soit q-stable sur un voisinage V de 0. Alors pour e dans V :

$$\left|\frac{A(d_0+e)-A(d_0)}{X}\right| \leq C.\left[\sup_{i=1..N}\left(\frac{|e_i|}{2^{\exp(x_i)}}\right)\right]^c$$

Sans restriction, nous pouvons prendre :

$$V=\prod_{i=1}^{N}\left[-2^{\exp(x_i)}.\varepsilon_0+x_i, 2^{\exp(x_i)}.\varepsilon_0+x_i\right]$$

Considérons maintenant (e_i) telle que si $j\neq i$, e_i et e_j soient indépendantes, et telle que e_i soit uniforme sur $[\,-2^{\exp(x_i)}\,\varepsilon, +2^{\exp(x_i)}\,\varepsilon\,]$, où $\varepsilon \leq \varepsilon_0$. On a alors :

$$\left|A(d_0+e)-A(d_0)\right|_{L_r^2} \leq C.\sqrt{E\left[\left(\sup_{i=1..N}\left|\frac{e_i}{2^{\exp(x_i)}}\right|\right)^{2q}\right]}$$

Les variables $\frac{|e_i|}{2^{\exp(x_i)}}$ étant indépendantes et toutes uniformément distribuées sur $[0,\varepsilon]$, la densité f de $\sup_{i=1..N}\left(\frac{|e_i|}{2^{\exp(x_i)}}\right)_{i=1..n}$ est donnée par :

$$f(x)=\frac{N.x^{N-1}}{\varepsilon^N}\cdot 1_{[0,\varepsilon]}(x)$$

On a donc, par suite :

$$E\left[\left(\sup_{i=1..N}\left|\frac{e_i}{2^{\exp(x_i)}}\right|\right)^{2q}\right] = \int_0^{\varepsilon}\frac{N.x^{2q+N-1}}{\varepsilon^N}.dx = \frac{N.\varepsilon^{2q}}{N+2q}$$

d'où

$$\left|A(d_0+e)-A(d_0)\right|_{L_r^2} \leq \sqrt{\frac{N}{N+2q}}\,C.\varepsilon^c$$

L'algorithme est donc q-stable par rapport à la perturbation p_d sur le voisinage $[0,\varepsilon_0]$. D'où :

$$q_d \geq q_p.$$

Par ailleurs, on a : $$C_d([0,\varepsilon_0], q) \leq \sqrt{\frac{N}{N+2q}} .C.$$

Il vient donc :

$$\text{Pour tout réel } q,\ C_d(q) \leq \sqrt{\frac{N}{N+2q}} .C_p(q)$$

La stabilité de l'algorithme machine par rapport aux perturbations p_e est donc généralement plus forte que la stabilité du problème. Cependant, la propriété suivante montre que des conditions assez larges peuvent assurer l'égalité.

Propriété.

Si de plus, $\forall\ q>q_p$ $\lim_{e\to 0} \dfrac{|A(x+e)-A(x)|}{\left[\sup_{i=1..N}\left(\dfrac{|e_i|}{2^{\exp(x_i)}}\right)\right]^q} = +\infty$, alors q_d est égal à q_p.

Démonstration :

Elle est presque identique à la première. De l'hypothèse, il vient en effet pour tout $q \geq q_p$ et tout C; pour (e_i) telle que si $j\neq i$ e_i, e_j soient indépendantes et telle que e_i soit uniforme sur $[\ -2^{\exp(x_i)}\varepsilon,\ +2^{\exp(x_i)}\varepsilon\ ]$, où $\varepsilon \leq \varepsilon_0$:

$$\left|A(d_0+e)-A(d_0)\right|_{L_r^2} \geq \sqrt{\frac{N}{N+2q}}\ C$$

ce qui implique que q_p est égal à q_d

On peut donc sous certaines hypothèses estimer expérimentalement la régularité du problème. Le théorème suivant indique maintenant le lien entre la stabilité par rapport aux perturbations stochastiques au dernier bit et la stabilité arithmétique pour la norme L_r^2.

Théorème sur la stabilité arithmétique :

Si $n\to\infty$, et si l'algorithme est arithmétiquement q_a stable à la précision machine (sur le voisinage $[-\varepsilon_m+\varepsilon_m]^n$) alors :

$$2q_s > q_a$$
$$C_s(q) \leq \varepsilon_m^{q/2}.\ C_a(q)$$

démonstration :

Supposons donc l'algorithme A auquel on s'intéresse arithmétiquement q stable dans un voisinage de $[-\varepsilon_m+\varepsilon_m]^n$ pour la norme L_2 des perturbations et la norme L_2^r sur les résultats. Soient alors (e_i) des erreurs arithmétiques dans ce voisinage. L'erreur entre la solution x(e) obtenue avec les (e_i) et la solution x obtenue sans perturbation vérifie alors :

$$|x(e)-x| \leq C.\left[\frac{\sum_{i=0}^{n}|e_i|^2}{n}\right]^{\frac{q}{2}}$$

Ceci est en particulier vrai pour toutes les réalisations de variables aléatoires (e_i) telles que :

$$\text{prob}\ (e_i = \varepsilon_m) = \text{prob}\ (e_i = -\varepsilon_m) = \frac{\alpha}{2}\ ,\ \text{prob}\ (e_i = 0) = 1-\ \alpha.$$

Et donc pour x(e) résultat de $p_s(\alpha)$

Or, sous ces hypothèses, $Y=\dfrac{\sum_{i=1}^{n}|e_i|^2}{\varepsilon_m^2}$ suit une loi binomiale de paramètre α.

Par suite : $E\left[\sum_{i=0}^{n}|e_i|^2\right]^q=\left[\left(\alpha.\varepsilon_m n\right)^q+o(n^q)\right]$. Asymptotiquement en n (si n est assez grand) on a donc :

$$\text{Pour tout réel q, } C_s(q) \leq \varepsilon_m{}^{q/2}.\ C_a(q)$$

$$q_s \geq q_a/2$$

Remarque : L'utilisation du théorème suppose donc que le nombre d'arrondis est grand.

Il reste maintenant simplement à lier la stabilité arithmétique à la régularité par rapport à la perturbation "précision paramétrée" p_m. Ce problème à déjà été traité au chapitre 3. La perturbation p_m étant un cas particulier de la perturbation arithmétique, on a :

$$q_m \geq q_a$$

$$\text{Pour tout réel q, } C_m(q) \leq C_a(q)$$

Nous venons donc de voir comment le calcul de la régularité d'un logiciel par rapport à des perturbations définies dans le chapitre 5 pouvait permettre d'avoir des indications des qualités arithmétiques de celui-ci. Nous avons également vu précédemment que sous certaines hypothèses, la régularité et le conditionnement d'un algorithme par rapport à une perturbation accessible en machine étaient estimables grâce à la régression loglinéaire. On regroupe maintenant ces deux outils pour proposer des indicateurs expérimentaux de stabilité.

B-5 Estimations de stabilité de la méthode SCALP.

On propose les estimateurs de stabilité suivants :

Estimation de la régularité et du conditionnement du problème :

Pour différentes valeurs de m (m_i) "assez petites "
pour différents échantillons
perturber les mj derniers bits des données uniformément
évaluer l'erreur (relative quadratique moyenne) sur le résultat
(la référence étant le résultat non perturbé) Ej
Calculer les coefficients (a,b) de la régression de log(Ej) sur mj
$q_p = b/\log(2)$

$$C_p(q_p) = e^{a} * (2^{-L})^{q_d} * \sqrt{\frac{N+2q_p}{N}}$$

Estimation de la stabilité de l'algorithme par perturbation stochastique :

Pour différentes probabilités de perturbation (α_j)
pour différents échantillons
perturber l'algorithme par $p_S(\alpha_j)$
évaluer l'erreur quadratique moyenne sur le résultat
(la référence étant l'arrondi au plus près) Ej
Calculer les coefficients (a,b) de la régression de log(Ej) sur α_j
$q_a = 2*b$
$C_a(q_a) = e^{a} * 2^{L}$

Estimation de la stabilité de l'algorithme grâce à la précision paramétrée :

Pour différentes valeurs du nombre de bits d'arrondis en moins (m_j)

exécuter l'algorithme en arrondi sur m_j bits

évaluer l'erreur sur le résultat

(la référence étant l'arrondi au plus près) E_j

Effectuer une régression linéaire de($\log(E_j)$)sur (m_j)

$q_a = b/\log(2)$

$C_a(q_a) = e^{a} * (2^{-L})^{q_a}$

C - EXEMPLES D'APPLICATIONS.

Nous allons maintenant montrer certaines possibilités de cette méthode, et donc de notre théorie de la stabilité sur quelques exemples de natures très différentes.

Nous verrons d'abord que pour un calcul de somme tout à fait simple pour lequel nous pouvons justifier la régularité 1, nos estimateurs sont capables de redonner cette régularité.

Nous étudierons ensuite le problème classique de résolution d'un système linéaire, par l'algorithme de décomposition QR utilisant la méthode de Householder, ou par l'algorithme de Gauss. Pour un tel calcul déjà complexe les régularités nous sont inconnues. Comment pouvons nous alors juger des performances de nos estimateurs ? Nous nous contenterons, d'une part de visualiser la bonne validité apparente des régressions, d'autre part de montrer comment nos estimateurs indiquent que l'algorithme QR est plus stable que l'algorithme de Gauss.

Nous regarderons ensuite succintement sur un exemple particulier deux algorithmes d'estimation récursive, l'algorithme LMS et un de ses dérivés LMS' supposé plus stable [Cio], [Com], sans qu'aucune démonstration n'existe à notre connaissance. Nous verrons que sur cet exemple la meilleure stabilité de LMS' est visualisée par nos estimateurs.

Enfin, nous décrirons rapidement une application à un logiciel industriel de notre méthode d'analyse de la stabilité sur un algorithme d'extrapolation analytique de trajectoire, du CNES.

C-1 Le cas d'une simple sommation.

Etudions d'abord le cas de la sommation en simple précision, par ordre décroissant des valeurs, des 30 premiers termes de la série de terme général $\frac{1}{3^i}$. Cette sommation a pour régularité 1. Nous avons donc étudié l'effet d'une arithmétique stochastique perturbée sur cet algorithme pour voir si nous pouvions retrouver cette stabilité par régression. La régression du logarithme de l'erreur quadratique moyenne par rapport à l'arrondi au plus près sur le logarithme de la fréquence des perturbations (variable t) visualisée maintenant redonne effectivement la régularité attendue.

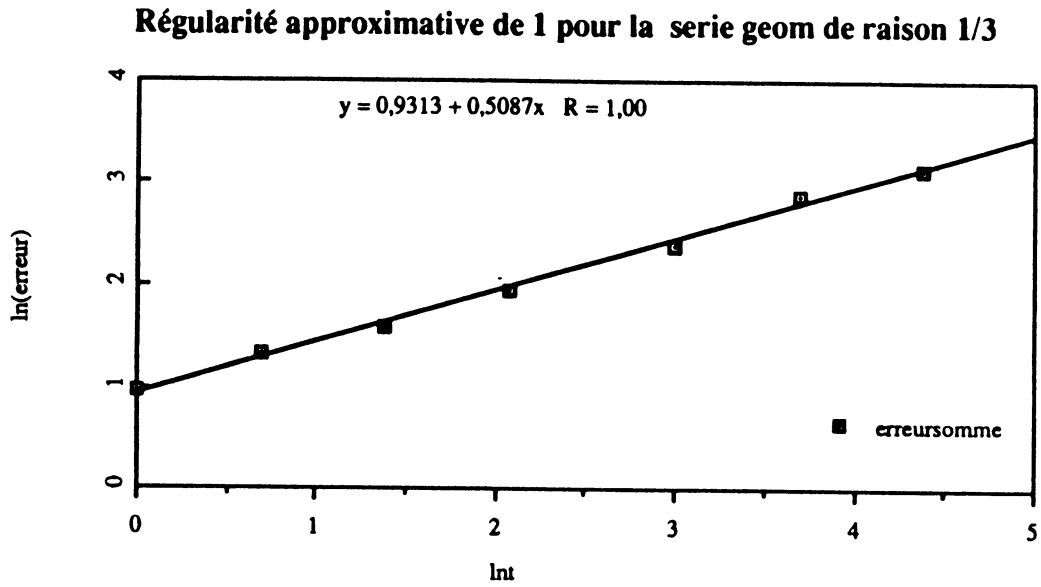


Figure III-1

Régularité estimée : $q_a = 2q_s = 0.5*2 = 1$

On visualise donc grâce à notre estimateur de régularité la bonne stabilité de la sommation étudiée.

C-2 Cas d'algorithmes matriciels classiques.

On étudie maintenant la résolution en simple précision d'un système linéaire (Ax = b) par l'algorithme de Gauss ou l'algorithme de factorisation QR (version Householder). A sera la matrice tridiagonale à diagonale dominante de type "matrice du potentiel" définie par :

$A_{ii} = +4$

$A_{ii+1} = A_{ii-1} = -1$

Toutes les composantes de b sont prises égales à 2 sauf la première et la dernière qui sont égales à 3. On prend un problème de taille 7. On cherche à mesurer la régularité de la quatrième composante qui a la plus forte erreur relative en moyenne (sur l'algorithme de Gauss et QR) lorsqu'on perturbe par la perturbation de Vignes [Che].

• On étudie d'abord les effets de perturbations arithmétiques stochastiques sur les algorithmes de Gauss et de factorisation QR en essayant de valider notre modèle de régression du deuxième paragraphe. Comme on l'a expliqué précédemment, on régresse l'erreur relative sur le résultat sur les logarithmes des probabilités (t) de perturbation (figures III-2, III-3).

Stabilité de l'Algorithme de GAUSS

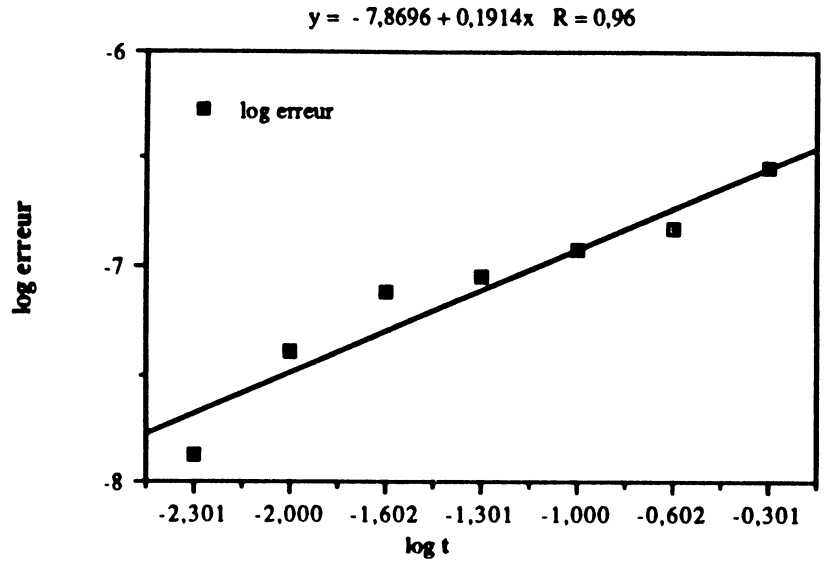


Figure III-2

Régularité estimée: $q_a = 2 * q_s = 0.38$

Stabilité de la décomposition QR

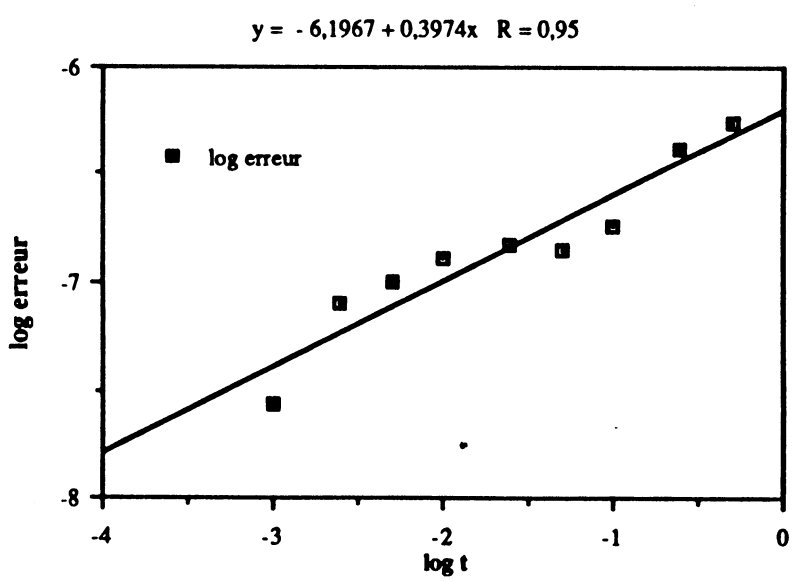


Figure III-3

Régularité estimée: $q_a = 0.397 * 2 = 0.80$

Les régressions paraissent licites. Les expériences sont donc en accord avec notre modèle de régularité. Mais que pouvons nous en tirer ? Dans ce cas précis, nous pouvons aussi relier les estimations de stabilité obtenues avec ce que nous connaissons sur les algorithmes de Gauss et de décomposition QR. En effet, nous visualisons bien par la méthode SCALP la meilleure stabilité notoire de la décomposition QR (régularité 0.38 pour la méthode de Gauss et 0.80 pour QR).

• Etudions maintenant la stabilité des problèmes machines par la méthode présentée antérieurement (figures III-4, III-5).

Algorithme de GAUSS (stabilité du problème)

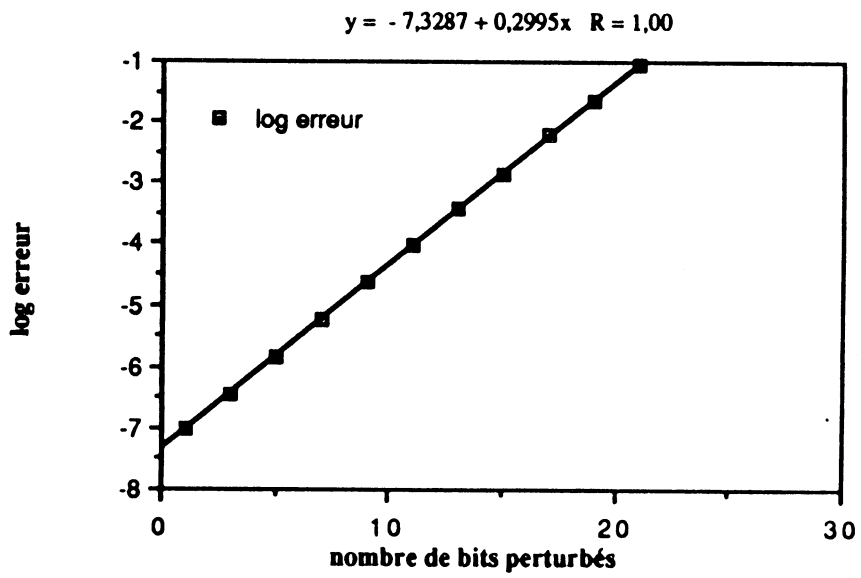


Figure III-4

Régularité estimée: $q_p = q_d = \frac{0.29}{\log(2)} = 0.99$

Décomposition QR (stabilité du problème)

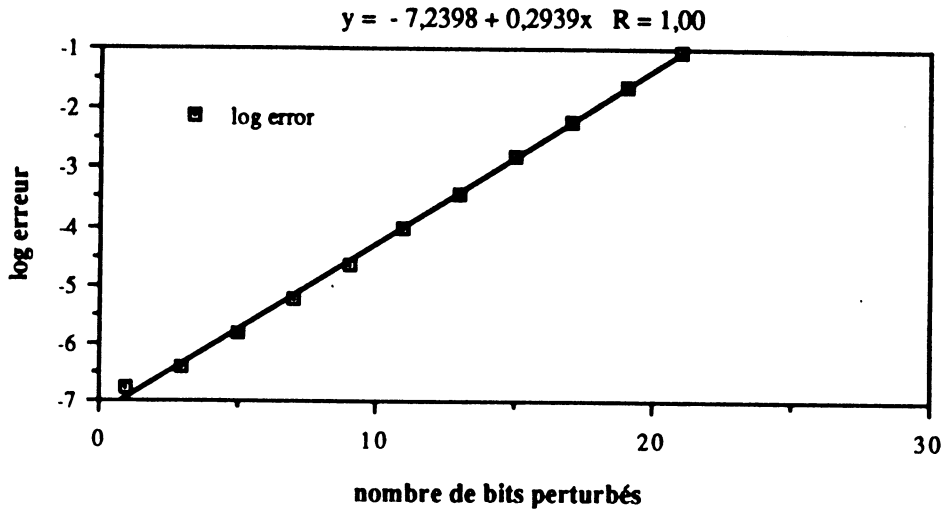


Figure III-5

Régularité estimée : $q_p = q_d = \frac{0.293}{\log(2)} = 0.\ 98.$

Les modèles de régression sont donc parfaitement justifiés dans les deux cas. De plus, la régularité estimée des deux problèmes machine est approximativement 1 tout comme celle du problème d'inversion (qui est lisse). Cela signifie que les deux problèmes machine étudiés représentent bien en un certain sens le problème d'inversion; un résultat qu'on pouvait attendre à cause du bon conditionnement de la matrice du système étudié.

Ainsi, pour une matrice bien conditionnée particulière notre modèle de régression est effectivement validé. De plus les régularités estimées nous donnent des résultats intéressants sur la qualité des algorithmes étudiés, notamment dans l'optique du choix de l'algorithme le plus stable. On notera aussi qu'une étude du seul problème machine ne nous permettrait pas, par contre, de trancher nettement en faveur de la méthode de Gauss ou de la décomposition QR.

Le modèle de régression n'est évidemment pas tout le temps aussi valide. Nous étudions à titre d'exemple la résolution par l'algorithme de Gauss du système linéaire de taille 7 ayant le même second membre que le système précédent mais dont la matrice est la matrice de Hilbert ($H_{ij} = \frac{1}{i+j-1}$). Le résultat des perturbations arithmétiques stochastiques est le suivant (figure III-6).

Stabilité de l'algorithme de Gauss sur la matrice de Hilbert de taille 7.

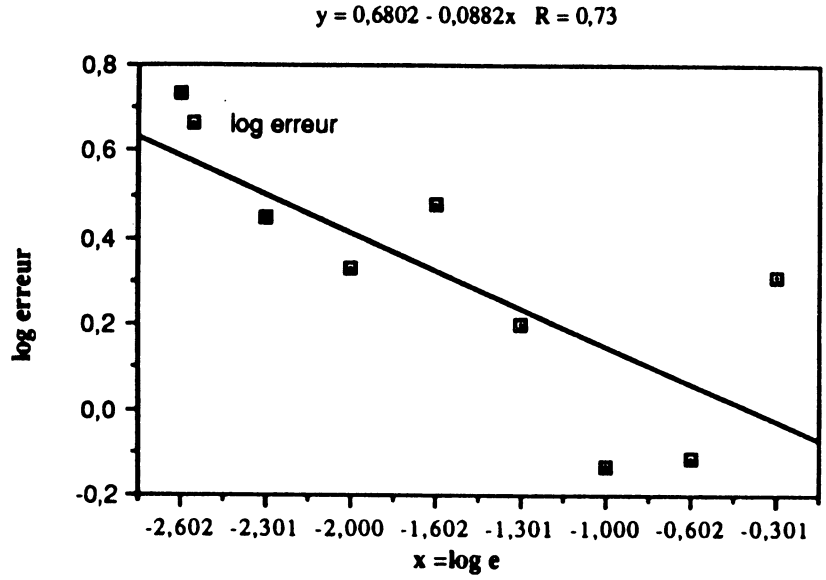


Figure III-6

Bien que le modèle de régression ne soit pas aussi valide, on remarque que notre estimateur indique bien l'instabilité de l'algorithme puisque la régularité estimée est négative. Cette instabilité est aussi visible par analyse du problème machine pour lequel la régression est de plus licite :

Instabilité du problème

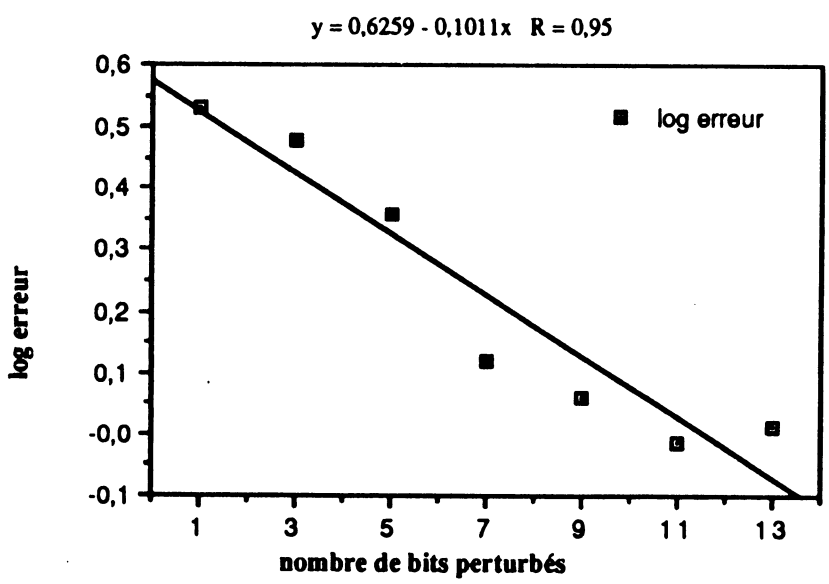


Figure III-7

C-3 Les algorithmes LMS.

Dans cette partie nous appliquons notre outil sur un exemple pour deux algorithmes de traitement du signal : les algorithmes LMS (Least Mean Squares) [Cio], [Com].

Etant donnés X_k et d_k des processus observés successivement (d_k scalaire et X_k vectoriel) le but de ces algorithmes est de construire une régression linéaire (W_k) de d_k sur X_k afin d'être capable d'estimer d_{k+1} sachant X_{k+1}et de faire de la prévision. L'algorithme LMS peut être écrit sous la forme :

$$\text{(LMS)} \qquad W_{k+1} = W_k + 2\mu X_k(d_k - X_k^t W_k),\ W_0 \quad (\mu \text{ suffisament petit})$$

où X_k^t désigne le transposé de X_k.

Pour cet algorithme des instabilités arithmétiques peuvent apparaître pour k assez grand [Cio], [Com]. Une version modifiée de LMS (empiriquement plus stable) a été proposée [Cio] :

$$\text{(LMS')} \qquad W_{k+1} = (1-2\mu a)\, W_k + 2\mu X_k(d_k - X_k^t W_k) \qquad a > 0 \ \text{(a suffisament petit)}$$

Remarque :

- La meilleure stabilité de (LMS') n'est, à notre connaissance, pas encore démontrée [Com]. C'est seulement une constatation expérimentale.

- Pour assurer la convergence numérique des algorithmes LMS', on choisit X_k tel que :

$\lim_{k\to\infty}(X_kX_k^t)$ existe et μ est assez petit relativement à $1/(\lambda_{max}+a)$, où λ_{max} est la plus grande valeur propre de $E\{\lim_{k\to\infty}(X_kX_k^t)\}$.

Nous avons validé nos estimateurs pour ces algorithmes sur des données particulières et avons retrouvé la meilleure qualité de LMS'. Pour être capable d'assurer les conditions de convergence données antérieurement, nous avons pris (X_k) comme la suite des itérés successifs d'une méthode de Durand-Kerner de calcul des racines d'un polynôme. Nous donnons maintenant les mesures de régularité obtenues pour les deux algorithmes (figures III-8, III-9) :

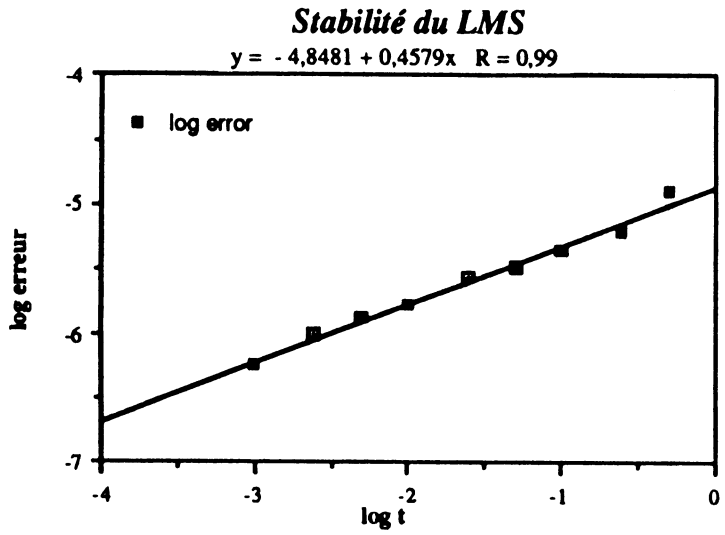


Figure III-8

Régularité estimée : $q_a = 2q_s = 2*0.457 = 0.91$

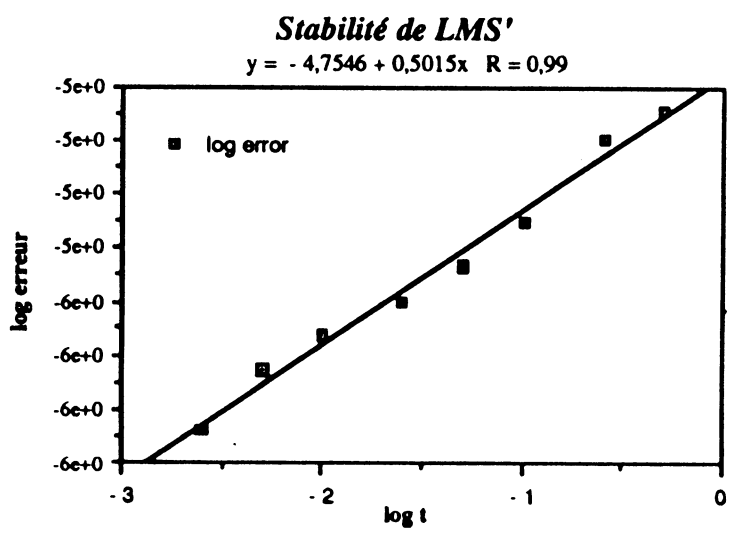


Figure III-9

Régularité estimée: $q_a = 2*0.5 = 1$

Nos estimateurs de stabilité sont donc ici aussi intéressants notamment dans la mesure où ils sont capables de mettre en évidence la meilleure stabilité du deuxième algorithme.

On présente maintenant rapidement les résultats (figure III-10) pour le problème machine (les données perturbées sont les X_k) associé à LMS (on a le même résultat pour LMS').

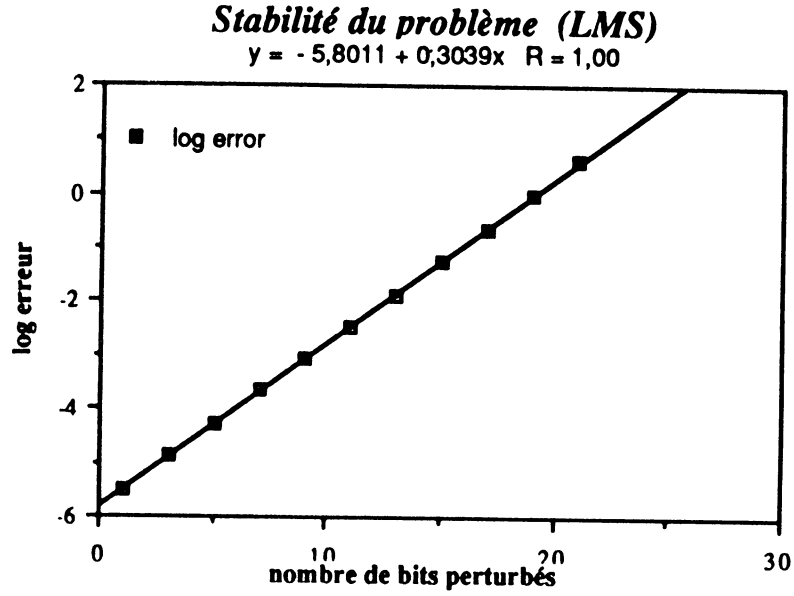


Figure III-10

Régularité estimée: $q_p = q_d = \frac{0.30}{\log(2)} = 1$

Ainsi, la régularité observée est de 1 et donc l'estimation récursive semble très stable par rapport à une perturbation de la trajectoire. Ainsi (même si les expériences ont été faites sur des données simples), nous pouvons aussi être satisfaits de nos estimateurs de stabilité sur les algorithmes LMS dans la mesure où les régressions sont valides et nous permettent de trouver la meilleure stabilité attendue de LMS'.

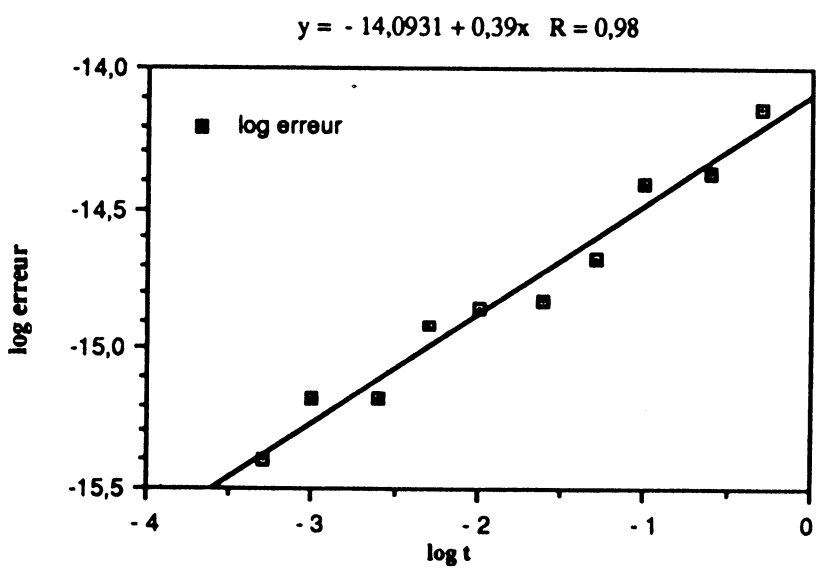
C-4 Le cas d'un logiciel industriel.

Comme nous l'avons signalé à de nombreuses reprises, ce travail s'inscrit dans le cadre d'une collaboration avec le Centre National d'Etudes Spatiales. Il était donc tout naturel de choisir parmi nos problèmes cibles un logiciel mis au point au CNES. Il a été choisi d'étudier le logiciel Maestro d'extrapolation analytique d'orbites. A partir d'un vecteur X_0 donnant la position et la vitesse initiale d'un satellite soumis au seul potentiel gravitationnel, ce produit permet d'extrapoler la position et la vitesse de l'engin $X_T = F(T,X_0)$ après une durée T. Deux types de méthodes existent pour effectuer un tel calcul. Les méthodes classiques consistent à intégrer itérativement (pas à pas) la trajectoire par des intégrateurs numériques (de type Runge-Kutta).

Les méthodes analytiques sont basées sur le calcul préliminaire par des méthodes de calcul formel d'une approximation formelle G de F, et consistent à effectuer directement le calcul $X_T = G(X_0)$. Les méthodes numériques, bien que considérées comme assez précises, présentent l'inconvénient d'être lentes dès que la durée d'intégration de la trajectoire est trop grande.

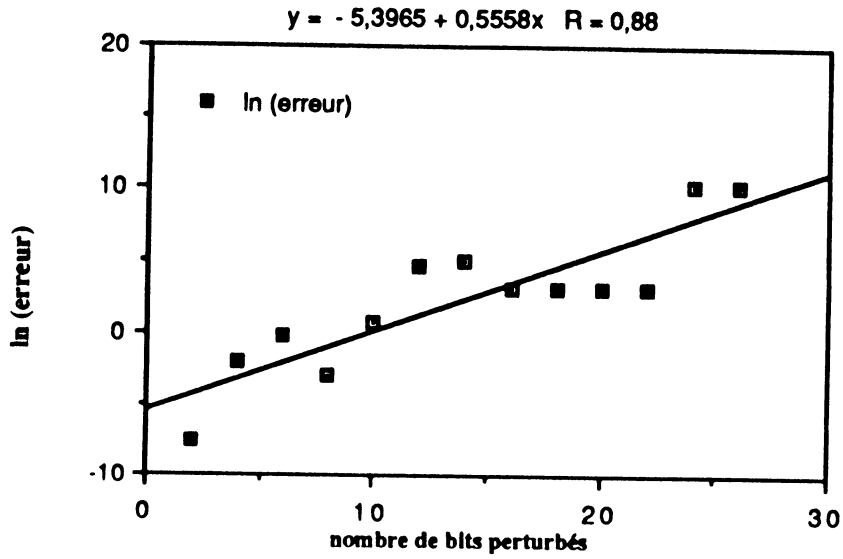
On est donc amené à utiliser des intégrateurs analytiques comme Maestro, et il faut étudier leur précision aussi bien numérique qu'arithmétique. Le logiciel est d'une grande complexité tant au niveau mathématique qu'au niveau du code Fortran (qui contient plus de 14000 lignes). Nous avons donc dû travailler sur ce produit avec très peu de connaissances *a priori* sur son fonctionnement, c'est à dire en le considérant pratiquement comme une "boite noire". On pourrait penser que la validation de nos estimateurs de stabilité sur un tel logiciel ne présente pas grand intérêt dans la mesure où on manque de références. Il nous parait au contraire intéressant de signaler que nos estimateurs de régression se sont avérés valides dans ce cas industriel. Nous donnons maintenant les résultats obtenus sur la deuxième composante de la vitesse en comparant en particulier (pour se donner une référence) les deux estimateurs de régularité de l'algorithme proposés dans la seconde partie.

Régularité de l'algorithme visualisée par perturbation stochastique



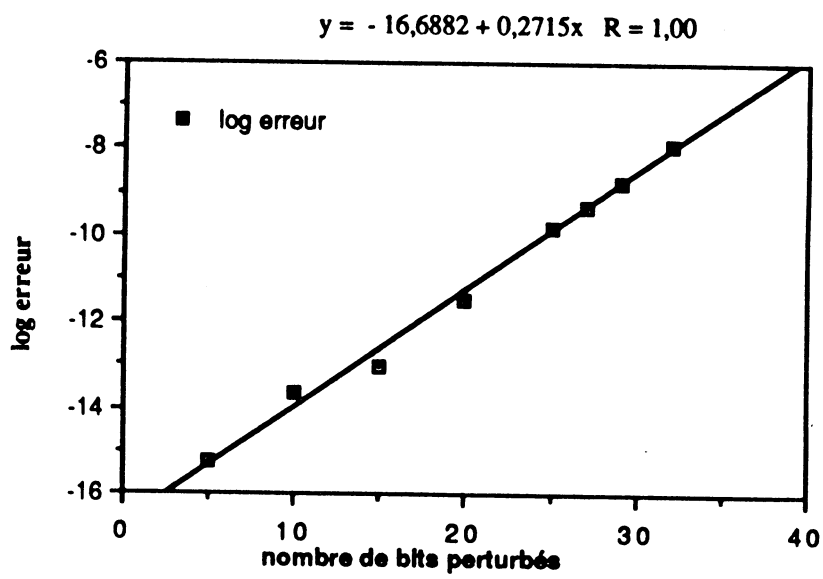
Régularité estimée : $q_a = 2*q_s = 0.39*2 = 0.78$

Régularité de l'algorithme visualisée grâce à la précision paramétrée



Régularité estimée : $q_a = q_m = \frac{0.55}{\ln(2)} = 0.79$

Régularité du problème



Régularité estimée : $q_p = q_d = \frac{0.27}{\log(2)} = 0.9$

Ainsi donc nos estimateurs visualisent une bonne stabilité de Maestro relativement au problème qu'il résout (une perte de régularité de seulement 0.1 par l'algorithme relativement modeste par rapport à la perte de régularité de 0.4 pour l'algorithme de Gauss sur la matrice de type potentiel) et un gain relativement fort de précision (0.8 bit) par bit de précision supplémentaire pour les calculs.

On remarquera également la bonne adéquation entre les deux estimateurs de régularité de l'algorithme qui confirme la validité d'un tel modèle dans ce cas.

D - CAS PARTICULIERS.

D-1 vers la validation d'hypothèses du modèle de Chesneaux.

Dans deux cas particuliers, le cas analytique et le cas d'un algorithme vérifiant les hypothèses du modèle de Chesneaux, nous pouvons prouver des résultats supplémentaires sur les résultats perturbés qui nous permettent de déboucher sur des heuristiques simples de test de ces cas. C'est l'objet de ce court paragraphe.

Nous noterons dans cette partie $p_{(t,q)}$ la perturbation stochastique au dernier bit de l'algorithme A définie au chapitre 4 associée à la perturbation des données :

$$p_{(t,q)}(x) = x + \alpha(t,q).2^{-L+exp(x)}$$

où :

$$\text{prob}\,(\alpha(t,q) = 1) = t, \quad \text{prob}\,(\alpha(t,q) = -1) = q$$

et où L désigne la longueur des mantisses et exp(x) l'exposant de x dans sa représentation virgule flottante. x(t,q) désignera le résultat de l'algorithme perturbé pour les probabilité de perturbation t et q; x désignera le résultat non perturbé.

On donne d'abord une propriété commune aux algorithmes linéaires et à ceux obéissant au modèle de Chesnaux :

Propriété 1.

Si l'algorithme est linéaire ou vérifie les hypothèses du modèle de Chesneaux, alors $E\left(x(\frac{t}{2}, \frac{t}{2}) - x\right)$ est indépendant de t.

Démonstration.

Dans ces deux cas, pour une perturbation arithmétique aléatoire de l'ordre du dernier bit, le résultat perturbé x(t,q) s'écrit [Che] :

$$x(t,q)=x'+\sum_{i=1}^{n} a_i e_i$$

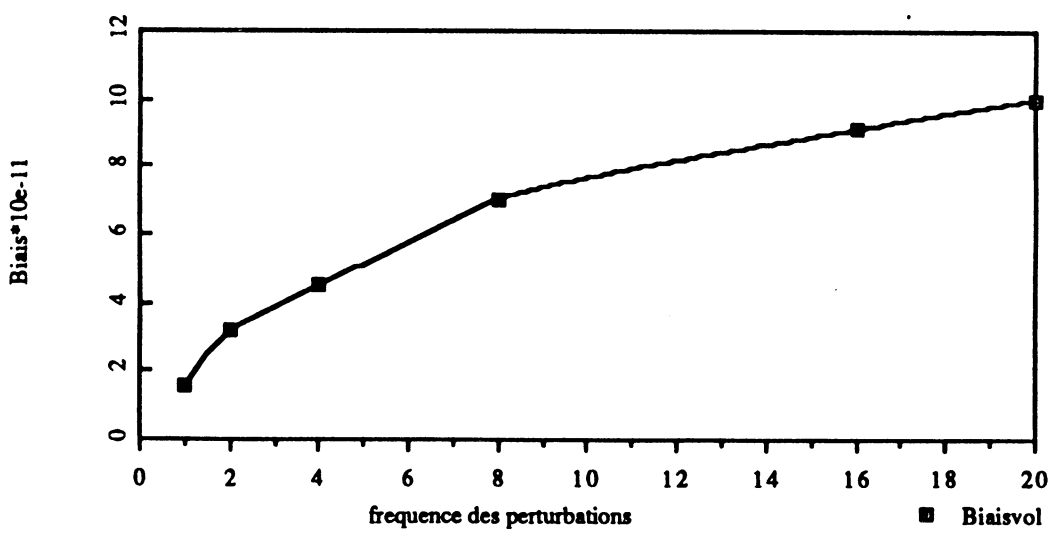
où $(e_i)_{i=1..N}$ est le paramètre qui définit la perturbation arithmétique considérée (cf chapitre III), c'est à dire le vecteur de toutes les perturbations des résultats intermédiaires, les a_i sont des constantes et où x' est égal au résultat obtenu en arrondi au plus près dans le cas linéaire (il a pour moyenne le résultat exact dans le cas du modèle de Chesnaux [Che], à cause du modèle de Feldstein invoqué dans ce modèle). Par suite, si e_i est une perturbation sans biais, ce qui est le cas lorsqu'on applique une perturbation $p(\frac{t}{2}, \frac{t}{2})$, $E\left(x(\frac{t}{2}, \frac{t}{2}) - x'\right)$ est une constante. D'où le résultat.

Ainsi, l'utilisation de perturbations stochastiques au dernier bit de différentes ampleurs peut aider à autovalider le modèle de Chesneaux ou à visualiser les non linéarités (C'est à dire la "volatilité" de l'algorithme (cf chapitre 3)).

Exemple.

On reprend l'exemple matriciel donné dans le deuxième chapitre pour invalider le modèle PRECIX. On visualise ci dessous la volatilité du résultat. C'est à dire l'évolution du biais x(t/2,t/2)-x en fonction de la probabilité de perturbation t.

Volatilité du calcul



Ainsi, l'importance des changements d'exposants ou des non linéarités dans l'algorithme de Gauss est vue grâce à l'observation de l'effet d'une augmentation de la variance des perturbations.

On donne maintenant une propriété qui, dans le même cadre, vise l'analyse des effets d'accumulation de biais locaux.

Propriété 2.

Sous les mêmes hypothèses, on peut écrire :

$$E\left(x(t,q) - x(\frac{1}{4},\frac{1}{4})\right) = (t - q).M$$

où M est un minorant des effets d'accumulation.

Démonstration :

Sous ces hypothèses, le résultat d'une perturbation arithmétique x(t,q) s'écrit en effet comme dans le cas de la propriété précédente :

$$x(t,q)=x'+\sum_{i=1}^{n} a_i.e_i$$

où les e_i sont les variables aléatoires représentant les arrondis, les a_i sont des constantes et où E(x') ne dépend pas de la perturbation.

Il vient alors

$$x(t,q)=E(x')+E\left(\sum_{i=1}^{n} a_i.e_i\right) = E(x(0.25,0.25)) + \sum_{i=1}^{n} a_i.E(e_i)$$

donc pour la perturbation $p_{(t,q)}$

$$E\left(x(t,q) - x(\frac{1}{4},\frac{1}{4})\right) = (t - q).M$$

où $M = 2^{-L} \cdot \sum_{i=1}^{n} a_i$ (L est le nombre de chiffres de mantisse dans la représentation virgule flottante)

est donc un minorant de la norme L_1 du vecteur d'accumulation : $2^{-L} \cdot \sum_{i=1}^{n} |a_i|$

Cette deuxième propriété signifie que pour des algorithmes presque linéaires, l'étude de l'évolution de $E(x(t,q)-x)$ avec p-q donnera un minorant des effets d'accumulation.

Remarque :

Ces deux propriétés ne fondent pas réellement une méthodologie à part pour les algorithmes analytiques mais donnent des outils supplémentaires d'analyse. C'est de plus sans doute une base intéressante pour valider automatiquement le modèle de Chesneaux [Che] et donc la méthode de Vignes [Che].

D-2 Utilisation de SCALP pour le calcul empirique d'erreurs : Une approche heuristique.

Le méthode SCALP se propose, comme nous l'avons vu, de donner sous certaines hypothèses une estimation de la régularité q et du conditionnement arithmétique C des algorithmes pour des normes naturelles.

Par définition, si X(e) est le résultat de la perturbation arithmétique pour $e = (e_i)$ assez petit et X le résultat non perturbé, nous avons :

$$|x(e)-x|_{L_r^2} \leq C \left[\frac{\sum_{i=0}^{n} |e_i|^2}{n} \right]^{\frac{q}{2}}$$

Or nous avons vu dans le chapitre III que le résultat de l'algorithme théorique était aussi, sous de larges hypothèses, le résultat d'une perturbation arithmétique de l'algorithme machine pour laquelle les e_i sont bornées en valeur absolue par le "macheps" (c'est à dire la différence entre 1 et son successeur immédiat en machine). Si l'algorithme machine est q stable dans le voisinage [-macheps, macheps], on peut donc écrire la relation ci dessus en remplaçant x(e) par le résultat exact R.

Mais par ailleurs, les modèles probabilistes globaux (voir chapitre II) nous disent que, dans ce cas, les e_i se comportent à peu près comme des échantillons d'une variable aléatoire uniforme sur [-macheps/2,macheps/2]. On peut donc écrire, sous l'hypothèse de la validité des modèles probabilistes globaux, et si n assez grand :

$$|x(e)-R|_{L_r^2} \leq C.\left[\frac{\varepsilon_m^2}{12}\right]^{q/2}$$

Mais par ailleurs, si l'on suppose avoir estimé q et C grâce à l'arithmétique stochastique et aux estimateurs donnés précédemment, on peut écrire en adoptant les mêmes notations qu'au paragraphe précédent :

$$|x(1/4,1/4)-x|_{L_r^2} \approx C.\left[\frac{\varepsilon_m^2}{2}\right]^{\frac{q}{2}}$$

Ce qui indique qu'une évaluation empirique de l'erreur d'arrondi peut être souvent obtenue par le calcul de l'erreur quadratique relative moyenne entre le résultat non perturbé et le résultat perturbé par la perturbation de Vignes. C'est l'évaluation empirique de l'erreur d'arrondi que nous avons retenu dans notre logiciel. La différence avec la méthode CESTAC est que nous prenons en compte le biais avec le résultat non perturbé.

E - CONCLUSION.

Dans ce chapitre, nous avons donc exposé et illustré notre méthodologie expérimentale SCALP de mesure de la stabilité. Nous ne prétendons pas à l'universalité. On peut évidemment trouver des exemples où nos estimateurs ne sont pas justifiés. Il reste aussi à étudier plus précisément les modalités pratiques (taille des échantillons...) d'efficacité de cette méthode. Cependant, les différents exemples ont, je l'espère, montré l'intérêt de l'approche pour valider des logiciels numériques.

D'un point de vue théorique il nous semble important de rappeler que le modèle des perturbations que nous utilisons est non paramétrique et n'est pas basé sur un développement du premier ordre qui, nous l'avons vu, est invalide dans certains cas. De plus, pratiquement, il faut se souvenir que SCALP est en un certain sens un outil qui s'autovalide.

Un outil perturbation apparait donc intéressant pour extraire d'un algorithme les informations qualitatives de la théorie du chapitre III. Dans le cas de problèmes mal posés (régularité négative) la détection de l'instabilité du calcul au sens quantitatif est déjà intéressante.

Cependant, certains problèmes mal posés vérifient des propriétés de stabilité qualitative. Dans un système chaotique qui a un attracteur étrange, par exemple, bien qu'il n'y ait aucune possibilité de localiser un point sur sa trajectoire dès qu'on perturbe (aussi faiblement soit-il) le système, l'attracteur de l'ensemble des trajectoires issues d'un bassin d'attraction est lui globalement stable pour des perturbations assez petites de la dynamique. Ruelle [Rue] précise ce phénomène à partir d'une définition d'un attracteur en terme de perturbations de la dynamique. Il serait sans doute intéressant d'étudier un apport possible des perturbations pour traiter ce type d'instabilité (de stabilité) [Rue] [Bru]. Nous rejoignons là F.Chatelin [Cha]. La théorie est encore à naître.

BIBLIOGRAPHIE

[Bru] MC BRUNET, Contribution à la fiabilité de logiciels numériques et à l'analyse de leur comportement, une approche statistique. Thèse de Doctorat, Université PARIS DAUPHINE, Janvier 1989.

[Cha] F CHATELIN, Résolution approchée d'équations sur ordinateur. Notes de cours de DEA de statistique, Université PARIS 6, 1989-1990.

[Che] J M CHESNEAUX, Etude théorique et implentation en ADA de la méthode CESTAC. Thèse de Doctorat de l'université PARIS 6, avril 1988.

[Dem] J W DEMMEL, On condition numbers and the distance to the nearest-ill posed problem number. Numer Math, 51, p251-289, 1987.

[Cio] J M CIOFFI, Limited precision effects in adaptative filtering. IEEE trans on circuits and systems, Vol cas-34, N°7, juin 1987.

[Com] P COMON, Etude de la stabilité numérique d'algorithmes adaptatifs en traitement du signal. Notes de travail, Thomson Sintra, mars 1988.

[Eco] P MAZODIER, Econométrie. Cours polycopié de 3ième année de l'ENSAE, Malakoff, 1982.

[Fle] R FLETCHER, Expected conditioning, IMA J of Num Analysis, p247-273, 5, 1985.

[Fra] P FRANCOIS, Mesure de l'erreur arithmétique et perturbations des calculs. Rapport de recherche IMAG-TIM3, Janvier 1989.

[Gol] J GOLUB, Matrix computations. John Hopkins Univ Press, Baltimore, 1983.

[Ric] J RICE, A theory of condition. SIAM J Numer. Anal. 3, pp287-310, 1966.

[Rue] D RUELLE, Small random perturbations of dynamical systems and definition of attractors. Comm in Math Phys, p137-151.

CONCLUSION GENERALE

"Je sais tout mais je n'y comprends rien."
R.Daumal.

Cette thèse étudie donc quelques problèmes liés à l'analyse de la stabilité arithmétique des logiciels numériques, notamment en essayant de préciser l'intérêt de l'utilisation de perturbations. Après avoir défini de nouveaux indicateurs généraux de stabilité dans le chapitre III, nous introduisons en particulier une méthodologie expérimentale "SCALP" d'estimation de ces indicateurs basée sur un lot d'arithmétiques perturbées.

Les trois principaux points caractérisant, d'un point de vue pratique, notre approche par rapport à d'autres outils basés sur des perturbations concernent :

- l'utilisation de différentes fréquences de perturbation pour révéler certaines non linéarités dans un logiciel (comme l'injection d'une incertitude dans un marché révèle la volatilité des prix).
- l'utilisation de deux niveaux de perturbation afin de qualifier un algorithme par rapport à la difficulté du problème qu'il résout.
- l'adoption, tout comme récemment F.Chatelin [Cha], d'une méthodologie d'analyse statistique des résultats ne faisant pas référence à un modèle statistique paramétrique mais seulement à des concepts d'analyse de la stabilité.

Nous disposons ainsi d'un outil d'analyse de la stabilité d'un programme, assez complet dans ses fonctionnalités.

D'un point de vue théorique, l'introduction dans le chapitre III d'un cadre général d'analyse de la stabilité permet, nous le croyons, de mieux préciser ce que l'on peut entendre par qualification du logiciel numérique et d'adopter un langage d'analyse des erreurs d'arrondi recoupant celui utilisé dans d'autres domaines.

De nombreuses questions posées par le contrôle de la stabilité restent encore ouvertes.

En ce qui concerne l'utilisation de méthodes de perturbation, il parait utile d'affiner l'étude des estimateurs de régularité que nous proposons et éventuellement de rechercher de nouveaux estimateurs. Il semble également intéressant, sur la base du travail de Ruelle [Rue] (proposant une définition d'un attracteur en terme de perturbations de la dynamique) ou, d'un point de vue plus épistémologique, de la méthode générale d'analyse d'un phénomène dégagée par Thom de sa théorie des catastrophes [Tho], d'étudier l'apport de perturbations pour déterminer des caractéristiques plus qualitatives associées à un calcul. C'est d'ailleurs ce qu'essaie de faire F.Chatelin. On pourrait par exemple commencer par étudier des calculs dont la dynamique sous-jacente correspond à une minimisation de gradient. On pourrait peut-être ainsi lier les résultats obtenus à la théorie des catastrophes élémentaires et éventuellement déboucher sur une "théorie thermodynamique" partielle du calcul qu'évoque Chatelin.

La détermination de méthodes de minimisation de l'erreur est également un sujet de recherche intéressant qui mériterait sans doute d'être plus exploré. Malgré les difficultés posées (cf Chapitre I), une approche utilisant le calcul symbolique pourrait en particulier être intéressante pour éliminer certaines instructions instables dans un programme. Dans certains cas, des techniques de filtrage pourraient éventuellement être envisagées pour incorporer dans un calcul des connaissances *a priori* sur le résultat.

L'étude des problèmes d'instabilité arithmétique liés à la parallélisation des algorithmes commence également à être d'actualité. On semble constater que l'augmentation du parallélisme conduit à amplifier les problèmes d'erreurs d'arrondi. Les versions parallèles de l'algorithme de factorisation QR par la méthodes de Jacobi, par exemple, semblent expérimentalement plus instables [Dao] que l'algorithme séquentiel. Est-ce essentiellement dû à l'augmentation du nombre de calculs effectifs parfois entraînée par la diminution du temps de calcul comme semblait le suggérer Sameh [Sam] ? Ce ne serait alors qu'une facette du vieux principe selon lequel "plus on fait de calculs plus on fait d'erreurs". Quel sera le rôle d'une éventuelle introduction d'indéterminisme lorsque l'on décentralisera le contrôle ? C'est un enjeu intéressant que d'éclairer ces questions.

D'un point de vue théorique enfin, nous pensons qu'il faudrait continuer à étudier les questions de stabilité arithmétique en s'inspirant d'autres cadres d'analyse de résistance à des perturbations. Nous soulignons cependant que se posera toujours le problème d'adéquation de modèles continus à des phénomènes discrets et que cela n'est pas (comme nous l'avons souligné au chapitre II) sans poser de difficultés. Le domaine du contrôle de l'erreur arithmétique s'apparente en ce point, sans doute, aux sciences expérimentales.

En conclusion, nous ne croyons pas à une méthode universelle de contrôle d'erreur. A notre avis un atelier de qualification du logiciel numérique devra tirer parti des différentes méthodologies existantes, chacune d'elles pouvant donner des renseignements complémentaires.

On peut par exemple songer à utiliser conjointement une arithmétique d'intervalles [KM] [Lan] pour déterminer une borne d'erreur sur les résultats et un système d'arithmétiques stochastiques pour effectuer une analyse expérimentale de la stabilité des algorithmes ou (en supposant les hypothèses du modèle de Chesneaux vérifiées, cf chapitre II) calculer l'estimation du nombre de chiffres significatifs fournie par CESTAC. Les méthodes d'intervalles ayant, en ce qui concerne l'erreur, l'avantage de donner un résultat déterministe.

BIBLIOGRAPHIE

[Cha] F CHATELIN, Analyse statistique de la qualité numérique et arithmétique de la résolution approchée d'équations par calcul sur ordinateur. Etude F. 133, centre scientifique IBM FRANCE, avril 1988.

[Dao] M DAOUDI, Etude de la complexité de la décomposition orthogonale d'une matrice sur plusieurs modèles d'architectures parallèles. Thèse de doctorat de l'INPG, mai 1989.

[Eke] I EKELAND, Le calcul, l'imprévu. Collection science ouverte, éditions du seuil, Paris, 1984.

[KM] U W.KULISCH and W L.MIRANKER, Computer arithmetic in theory and practice. Academic Press, New York, 1981.

[Lan] E IANGE, Implementation and test of the ACRITH facility in a system 370. IEEE trans on computers, vol C36, N°9, September 1987.

[PV] M La PORTE et J VIGNES, Error analysis in computing. Information processing, 1974.

[Sam] A SAMEH, Numerical parallel algorithms, a survey. In high speed computer and algorithms organization. D Kuck, D Lawrie and A Sameh eds, pp 207-228, Academic Press, 1977.

[Tho] R THOM, Modèles mathématiques de la morphogénèse. Editions Christian Bourgeois, Paris, 1981.

[Rue] D RUELLE, Small random perturbations of dynamical systems and definition of attractors. Comm in Math Phys, pp137-151.