# Imprécisions numériques : méthode d'estimation et de contrôle de la précision en C.A.O

Denise Pirus

# THÈSE

**Présentée à**

## l'UNIVERSITÉ de METZ

Pour l'obtention du grade de :
DOCTEUR de l'UNIVERSITÉ de METZ

## Spécialité : INFORMATIQUE

Denise PIRUS

# IMPRÉCISIONS NUMÉRIQUES : MÉTHODE D'ESTIMATION ET DE CONTRÔLE DE LA PRÉCISION EN C.A.O.



Soutenance à Metz le 2 juillet 1997

**Composition du jury :**

| | | |
|---|---|---|
| *Directeur de thèse* : | Yvon GARDAN | (Professeur à l'Université de Metz) |
| *Rapporteurs* : | Christian SAGUEZ | (Professeur à l'École Centrale de Paris, Directeur des Relations Industrielles et des Filiales, CNES, Paris) |
| | Michel VÉRON | (Professeur à l'Université de Nancy 1) |
| *Examinateurs* : | Georges RHIN | (Professeur à l'Université de Metz) |
| | Jean-Pierre JUNG | (Professeur à l'Université de Metz) |
| | Didier GALMICHE | (Habilité à diriger des recherches Nancy1 - CRIN) |

LABORATOIRE DE RECHERCHE EN INFORMATIQUE DE METZ

*Quand on travaille pour plaire aux autres on peut ne pas réussir,*
*mais les choses qu'on a faites pour se contenter soi-même*
*ont toujours une chance d'intéresser quelqu'un.*

*M. Proust*

## REMERCIEMENTS

Tout d'abord, je tiens à remercier mon directeur de thèse, Yvon Gardan, notamment pour la confiance qu'il m'a témoignée en m'acceptant dans son Laboratoire. Les conseils et les lignes directrices qu'il m'a proférés m'ont été précieux et m'ont permise de mener à terme ce travail. Je le remercie également pour le temps passé, malgré les nombreuses fonctions dont il a la charge, à lire et relire les diverses versions de ce manuscrit.

Je remercie tout particulièrement Messieurs Christian Saguez et Michel Véron pour avoir accepté d'être rapporteurs de ce travail, de même que Messieurs Georges Rhin, Jean-Pierre Jung et Didier Galmiche, pour leur participation à ce jury en tant qu'examinateurs.

Je tiens à exprimer mon amitié à l'ensemble du Laboratoire de Recherche en Informatique de Metz : *Ahmed, Benoît, Catherine, Cathy, Christian, (les) Dominique, Eric, Estelle, (les) Frédéric, Isabelle, Jocelyne, Kusno, Martine, Mickaël, Myriam, Pierre-Paul, Robin, Sandrine*, ...

J'ai aussi une pensée pour l'ex-quatrième étage (Département de Mathématiques), pour les agréables moments passés ensemble : *Jean-Marc, Souad, Véronique et Fabien, Nicolas, Pierre, Frédéric, Simone*, ...

A toutes ces personnes et à celles que je n'ai pas citées, mais à qui je pense, merci.

Denise

## SOMMAIRE

**Introduction.** ....1

**Chapitre I. État de l'art.**

1. Introduction. ....4
2. Problèmes associés à la C.A.O. ....4
3. La cause principale : l'arithmétique flottante. ....6
   - 3.1. Représentation des réels. ....6
   - 3.2. Erreur d'affectation. ....7
   - 3.3. Erreur sur une addition arithmétique. ....8
   - 3.4. Erreur sur une soustraction arithmétique. ....8
   - 3.5. Erreur sur une multiplication arithmétique. ....8
   - 3.6. Erreur sur une division arithmétique. ....9
   - 3.7. Programmation avertie. ....9
4. Approches de solutions au problème de l'imprécision numérique. ....10
   - 4.1. Outils mathématiques. ....10
     - 4.1.1. Analyse différentielle. ....10
     - 4.1.2. Conditionnement d'un problème. ....11
     - 4.1.3. Permutation-Perturbation. ....13
     - 4.1.4. La différentiation automatique. ....13
   - 4.2. Techniques de résolution. ....14
     - 4.2.1. Utilisation d'une tolérance. ....14
     - 4.2.2. Arithmétique rationnelle. ....15
     - 4.2.3. Arithmétique d'intervalle. ....19
     - 4.2.4. Représentation symbolique. ....20
     - 4.2.5. Modification des données symboliques. ....22
     - 4.2.6. Aquarels. ....23
5. Conclusion. ....24

**Chapitre II. Étude des erreurs commises au cours de l'intersection de deux droites**

1. Introduction. ....29
2. Présentation de l'algorithme d'intersection. ....29
3. Analyse différentielle. ....30
   - 3.1. Erreurs absolues. ....31

3.2. Erreurs relatives. ....32
3.3. Erreurs d'arrondi. ....33
3.4. Études expérimentales. ....34
3.5. Conclusion. ....36
4. Arithmétique d'intervalles. ....37
4.1. Erreurs théorique. ....37
4.2. Études expérimentales. ....39
4.3. Conclusion. ....43
5. Conclusion. ....43

**Chapitre III. Contrôle stochastique des arrondis de calcul**

1. Introduction. ....46
2. Propagation des erreurs d'arrondi dans un algorithme. ....46
3. Contrôle stochastique des erreurs d'arrondi. ....47
3.1. Introduction. ....47
3.2. La méthode de permutation-perturbation ou CESTAC. ....47
3.2.1. La méthode de permutation-perturbation : Aspect théorique. ....47
3.2.2. Aspect pratique de la méthode CESTAC. ....48
3.2.3. Estimation de la précision d'un résultat informatique. ....49
3.2.4. Validité, stabilité et robustesse de la méthode. ....50
3.3. Influence des erreurs de données sur les résultats d'algorithmes finis. ....51
3.4. Zéro informatique. ....52
3.5. Implémentation asynchrone de la méthode. ....52
3.6. Le problème des instructions conditionnelles. ....53
3.7. Implémentation synchrone. ....54
4. Exemples. ....54
5. Conclusion. ....56

**Chapitre IV. Calcul formel**

1. Introduction. ....58
2. Aspect général d'un système de calcul formel. ....58
2.1. Généralités. ....58
2.2. Les avantages. ....59
2.3. Les inconvénients. ....60
2.4. Conclusion. ....60
3. Le système Pari. ....61

3.1. Présentation du système Pari. ....61
3.2. Interface Pari-Modeleur. ....61
3.3. Conclusion. ....62
4. Alignement d'un flottant sur un rationnel. ....62
4.1. Introduction. ....62
4.2.Décodage de la repésentation binaire. ....63
4.3. Grille rationnelle. ....63
4.4. Méthode naïve d'approximation. ....63
4.5. Développement en fraction continue. ....64
4.6. Conclusion. ....66
5. Études expérimentales. ....66
5.1. Introduction. ....66
5.2. Étude 1. ....67
5.3. Étude 2. ....67
5.4. Étude 3. ....71
6. Conclusion. ....72

**Chapitre V. La méthode E.C.P en C.A.O**

1. Introduction. ....75
2. La méthode E.C.P. ....75
2.1. Introduction. ....75
2.2. Initialisation des données. ....75
2.3. Calcul du nombre de chiffres significatifs lors des calculs. ....76
2.4. Utilisation de l'arithmétique rationnelle. ....76
3. Mise en oeuvre. ....77
3.1. Introduction. ....77
3.2. Nouvelle définition du réel. ....77
3.2.1. Formulation du problème. ....77
3.2.1. Structure du réel. ....78
3.3. Nouveau concept de décidabilité. ....78
3.3.1. Introduction. ....78
3.3.2. Les instructions conditionnelles. ....78
a) Test de nullité. ....79
b) Tests de comparaison. ....80
3.3.3. Caractérisation des opérateurs de comparaison. ....80
4. Exemples. ....81
5. Conclusion. ....83

**Conclusion**. ..... 85

**Annexes**

Annexe 1 : Algorithme de Bentley-Ottmann. ..... 90
Annexe 2 : Algorithme d'intersection de droites ..... 92
Annexe 3 : Calcul de l'abscisse x. ..... 94
Annexe 4 : Calcul de l'ordonnée y. ..... 97
Annexe 5 : Calcul du déterminant. ..... 98
Annexe 6 : Calcul de d1. ..... 99
Annexe 7 : Algorithmes d'intersection de deux droitres ..... 100
Annexe 8 : Théorème 1. ..... 102
Annexe 9 : Algorithmes de la méthode CESTAC. ..... 103
Annexe 10 : Théorème 2. ..... 107
Annexe 11 : Théorème 3. ..... 108
Annexe 12 : Algorithmes de la méthode E.C.P. ..... 110
Annexe 13 : Nouveaux tests. ..... 112

**Références bibliographiques**. ..... 114

## INTRODUCTION

Dans la réalisation d'un système de C.A.O (Conception Assistée par Ordinateur), on se heurte fréquemment aux problèmes de précisions numériques et de décidabilité. Une des causes principales de ces complications est la représentation des nombres en virgule flottante. Cette représentation engendre des erreurs d'arrondi qui se propagent au cours des algorithmes, et on assiste à une dégradation progressive de la précision des résultats.

L'objectif de ce mémoire est d'apporter une solution permettant de maîtriser, voire diminuer, ces erreurs numériques et d'estimer la précision des résultats, en particulier pour les tests de décidabilité.

Nous rappelons dans le chapitre un, quelques notions sur l'arithmétique des ordinateurs en virgule flottante et les problèmes qu'elle peut entraîner, en particulier dans les domaines de la C.A.O. Ensuite, nous analysons les diverses solutions proposées dans la littérature au cours de ces dernières années. Nous essayons de montrer les avantages et inconvénients de chaque méthode.

Pour mettre en évidence et pour étudier la propagation des erreurs au cours des algorithmes, nous étudions le cas particulier de l'intersection de deux droites (ou segments). Cette étude se compose de deux parties :

La première traite l'étude de la propagation des erreurs à l'aide d'une analyse différentielle. Nous déterminons ainsi une expression des erreurs absolues (resp. relatives) affectant les coordonnées du point d'intersection en fonction des erreurs absolues (resp. relatives) liées aux données initiales. Nous appliquons cette étude à un exemple qui montre que cette analyse ne permet pas une estimation convenable des erreurs affectant les résultats d'un point de vue informatique.

Cette constatation nous conduit à l'utilisation de l'arithmétique d'intervalle, qui compose la deuxième partie. Nous appliquons cette arithmétique au cas particulier d'intersection de droites, en considérant qu'un réel est représenté par un intervalle dépendant de l'erreur relative de la machine. Nous obtenons alors des intervalles contenant les coordonnées du point d'intersection. Nous effectuons une étude statistique qui nous permet de déterminer une majoration de l'erreur relative perdue en fonction de l'angle entre les deux droites. Nous constatons cependant, que cette méthode n'est pas facilement généralisable et qu'elle ne permet pas d'éviter ou de minimiser les erreurs effectuées par l'ordinateur.

Pour estimer la précision d'un résultat informatique, nous présentons dans le troisième chapitre, la méthode CESTAC (Contrôle STochastique des Arrondis de Calculs) développée par J. Vignes et M. Pichat [VIG 93]. Cette méthode permet d'estimer le nombre de chiffres

significatifs affectant tout résultat d'un calcul numérique. Elle détecte au cours d'un programme si les calculs s'effectuent correctement ou non. La notion de "zéro informatique" y est développée et permet une nouvelle conception de décidabilité dans le débranchement au cours des algorithmes.

Si l'estimation de l'erreur effectuée par la méthode CESTAC est critique, nous voulons réduire ces erreurs. Pour cela, nous proposons de recourir à une arithmétique exacte dans les cas critiques, c'est à dire lorsque l'on estime que la précision des résultats est insuffisante.

Cependant, l'arithmétique rationnelle nécessite l'utilisation des grands entiers et une arithmétique spéciale. Afin de résoudre ce problème, nous étudions dans le quatrième chapitre la possibilité d'utiliser une interface entre le système SACADO (Système Adaptatif de Conception et d'Aide au Développement par Ordinateur), logiciel développé au Laboratoire de Recherche de Metz, et un logiciel de calcul formel. Nous avons retenu le logiciel de calcul formel PARI [BCO 95], car il est plus rapide dans les calculs arithmétiques que les autres logiciels et parce que nous pouvons l'utiliser comme une bibliothèque de fonctions arithmétiques en C. Nous décrivons aussi les diverses méthodes permettant le passage d'un nombre en virgule flottante à un nombre rationnel.

Le chapitre cinq présente notre méthodologie, qui permet à l'algorithme de choisir entre l'arithmétique à virgule flottante ou l'arithmétique exacte afin que les résultats aient le plus de chiffres significatifs possibles. Le critère du choix de l'arithmétique se fait à l'aide de l'estimation du nombre de chiffres significatifs des résultats des calculs que permet d'effectuer la méthode CESTAC .

Cette méthode permet donc de faire une estimation quantitative de la propagation des erreurs d'arrondi, et si cette propagation s'avère trop importante, d'effectuer les calculs de façon exacte. On obtient ainsi une précision maximale en évitant les problèmes introduits par l'arithmétique en virgule flottante qui ont été présentés au chapitre un.

Dans les calculs numériques des logiciels de C.A.O., les tests de comparaison et d'égalité sont en général réalisés à l'aide d'une précision dépendant des objets traités, des dimensions de l'espace dans lequel on travaille. Le choix de cette tolérance est délicat, et la valeur qu'on lui attribue n'est pas toujours fiable, car elle ne tient pas compte des erreurs cumulées. Aussi, l'implémentation synchrone de la méthode CESTAC apporte un concept différent de décidabilité dans les tests de comparaison ou d'égalité. L'utilisation d'une précision dans les tests comparatifs s'avère inutile, les comparaisons sont réalisées par rapport aux précisions des résultats.

Nous terminons ce travail, en appliquant cette méthodologie à divers exemples, afin de valider les différents concepts.

# CHAPITRE I

# ÉTAT DE L'ART

## 1. INTRODUCTION

Les incohérences numériques sont principalement dues à l'imprécision numérique qui entache tout calcul en précision finie. Les données géométriques (réels) sont généralement représentées en machine par des nombres en virgule flottante qui les approchent. Or les algorithmes géométriques sont amenés à comparer des éléments (points, droites, ...) qui peuvent être très proches, et le risque de conclure (par exemple) que deux points distincts p et q soient identiques sur la base d'un simple calcul flottant sur les coordonnées est permanent.

Ce chapitre regroupe les principaux outils mathématiques permettant d'étudier l'influence des erreurs et les principales méthodes, suggérées dans la littérature, qui tentent de résoudre les problèmes causés par l'arithmétique flottante.

Au préalable, on présente quelques problèmes associés à la C.A.O dus principalement à la représentation des réels.

## 2. PROBLÈME ASSOCIÉS À LA C.A.O.

Lors des conceptions d'objets géométriques, on se confronte à différentes erreurs topologiques et numériques. On présente ici quelques exemples typiques de la C.A.O.

- Dans le calcul d'un point d'intersection. Si P est le point d'intersection de trois droites $D_1$, $D_2$, $D_3$ alors si $D_1 \cap D_2 = P_1$ et si $D_2 \cap D_3 = P_2$, on a en général $P \neq P_1 \neq P_2$.

- Violation de la topologie :

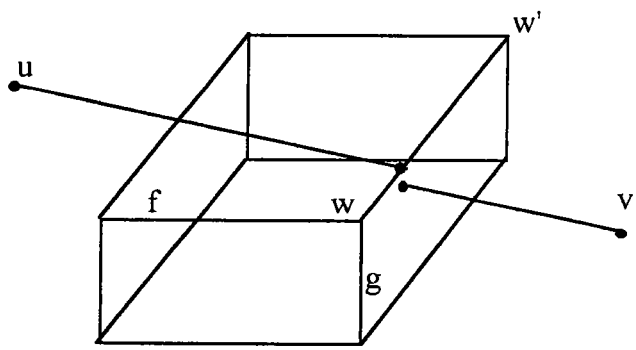



Figure 1

Si l'on se place dans une configuration comme celle suggérée par la figure 1, pour calculer l'intersection de l'arête [u,v] avec le polyèdre, on calcule l'intersection de cette arête avec chaque face de celui-ci. Si l'angle entre l'arête et la face f est très petit, le risque d'erreurs de calcul durant la détermination du point d'intersection est important, et on peut être amené à conclure qu'il est suffisamment proche de l'arête [w,w'] pour considérer qu'il se situe sur celle-

ci. Mais, quand on détermine le point d'intersection entre l'arête et la face g, la précision des calculs étant nettement meilleure on peut être conduit à conclure que le point n'appartient pas à l'arête [w,w']. Cette contradiction constitue une violation de la topologie, qui risque de causer des difficultés ultérieurement.

- Pour illustrer facilement le problème de précision numérique durant les calculs, on considère un pentagone A dans le plan, les points d'intersections de ses diagonales définissent un nouveau pentagone B qui se situe dans A. On appelle ce passage de A à B "going in" [HOF 89] et on note B = in(A). De même, si on prolonge les côtés de A, leurs intersections forment un pentagone C contenant A. On appelle cette opération "going out", et on note C = out(A). Il est facile de voir que A = out(in(A)) = in(out(A)). En effectuant n fois l'opération "going in", on obtient un pentagone B = $in^n$(A), on calcule alors C = $out^n$(B). On devrait théoriquement obtenir C = A, mais, si on compare les valeurs des coordonnées des sommets de A et C, on remarque que les erreurs introduites par les calculs augmentent rapidement, même pour des valeurs petites de n comme 2 et 3.

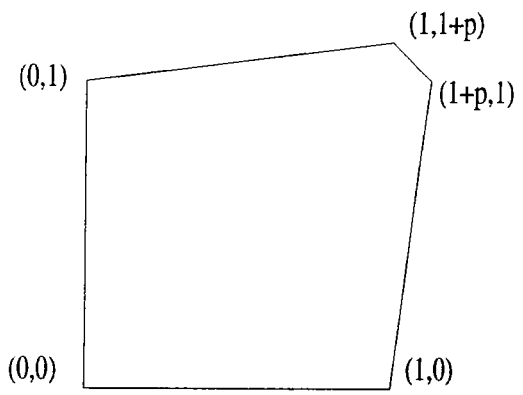



Figure 2 : Pentagone pour l'illustration des opérations "going in" et "going out".

Pour illustrer cet exemple, considérons le pentagone (Figure 2) de sommets (0,0), (1,0), (1+p,1), (1,1+p) et (0,1), pour quelques valeurs de n et p. Les test effectués (Tableau 1 et 2) avec des nombres à virgule flottante d'une précision de 11 décimales, démontrent facilement que les résultats deviennent rapidement imprécis, même pour des calculs simples (seule l'erreur absolue maximale obtenue sur les coordonnées est représentée).

Tableau 1 : Erreurs absolues maximales pour les opérations "going in", "going out".

| p | $out^1(in^1())$ | $out^2(in^2())$ | $out^3(in^3())$ | $out^4(in^4())$ |
|---|---|---|---|---|
| 0,1 | 1,82E-12 | 3,90E-10 | 5,88E-09 | 5,21E-07 |
| 0,01 | 0 | 4,34E-08 | 6,46E-06 | 3,05E-03 |
| 0,001 | 0 | 1,59E-05 | 3,64E-03 | 1,04E+00 |
| 0,0001 | 9,09E-09 | 1,79E-04 | 3,60E-03 | 1,00E00 |
| 0,00001 | 0 | 7,20E-03 | 4,8E-03 | |

Tableau 2 : Erreurs absolues maximales pour les opérations "going out", "going in".

| p | $in^1(out^1())$ | $in^2(out^2())$ | $in^3(out^3())$ | $in^4(out^4())$ |
|---|---|---|---|---|
| 0,1 | 3,64E-12 | 2,00E-11 | 5,24E-10 | 7,64E-09 |
| 0,01 | 1,82E-12 | 7,55E-10 | 1,31E-06 | 4,95E-04 |
| 0,001 | 1,82E-12 | 5,56E-09 | 9,88E-04 | 3,67E+00 |
| 0,0001 | 1,82E-12 | 2,24E-05 | 1,5E+00 | 5,02E+03 |
| 0,00001 | 0 | 1,86E-04 | | |

Les tableaux 1 et 2 illustrent les erreurs absolues commises par l'ordinateur, obtenues par différence entre les coordonnées calculées et les coordonnées théoriques. Nous voyons aisément que l'erreur grandit quand n augmente, car il est effectué plus d'opérations et il y a donc plus d'accumulation d'erreurs, et aussi quand la valeur de p diminue. En effet, pour des valeurs de p petites, le risque de perte d'informations (nombre de bits significatifs) augmente.

Ces différents problèmes sont causés principalement par l'arithmétique à virgule flottante.

## 3. LA CAUSE PRINCIPALE : L'ARITHMETIQUE A VIRGULE FLOTTANTE

### 3.1. Représentation des réels

Représentation théorique :

Le format mathématique pour représenter une valeur $x \in \mathbf{R}$ est :

$$x = \delta m b^e \quad \text{avec } \frac{1}{b} \leq m < 1$$

$\delta = \pm 1$ est le signe de x,

m est la mantisse illimitée (nombre infini de chiffres),

b la base,

e l'exposant entier signé.

Représentation en machine :

En machine, tout $x \in \mathbf{R}$, est représenté par $X \in \mathbf{F}$, $\mathbf{F}$ étant l'ensemble des nombres représentables en machine, tel que :

$$X = \delta M b^E$$

M est la mantisse codée sur t chiffres en base b (nombre fini de chiffres) et on admet que E = e.

On définit l'erreur relative d'arrondi par :

$$\gamma = \frac{x - X}{X} = \frac{m - M}{M} = \frac{r}{M}$$

r étant l'erreur absolue faite sur la mantisse m.

### 3.2. Erreur d'affectation

Dans la base b = 2, l'erreur absolue entre la valeur X et x est définie par :

$$X - x = -2^{E-t}\delta\alpha$$

La valeur $2^{-t}$ correspond à l'erreur absolue faite sur la mantisse M, t étant le nombre de bits de cette mantisse ($\delta$ représentant le signe de X).

Feldstein et Goodman [FEL 76] ont montré que dans les conditions de répartition logarithmique des mantisses, $\alpha$ est uniformément répartie sur [0, 1] en arithmétique tronquée et sur $\left[-\frac{1}{2}, \frac{1}{2}\right]$ en arithmétique arrondie au plus près.

Exemples d'erreurs introduit par les erreurs d'affectation :

- 25,47891 est représenté dans la base b = 10 avec une précision t = 7 par $0{,}2547891.10^2$. Avec t = 5 on a $0{,}25478.10^2$. Ce qui entraîne une erreur absolue de $0{,}91.10^{-3}$.

- Erreurs d'arrondi: Même si deux nombres a et b représentent exactement les nombres réels, leur produit peut être inexact.

Ex: Si a = 0,24045 et b = 0,62195 avec une précision p = 5 digits, on a a×b = 0,149547877, qui sera arrondi à 0,14955, ce qui entraîne une erreur d'arrondi de $0{,}2123.10^{-6}$.

- Erreurs d'annulation de digits : La différence entre deux nombres proches entraîne une perte de chiffres significatifs comme le montre M. Daniel [DAN 89]. Si on considère deux réels avec un nombre cmax de chiffres significatifs proches l'un de l'autre et si on note cc le nombre de chiffres significatifs égaux : $a = 0{,}c_1c_2...c_{cc}ab..\ 10^{\alpha}$ et $b = 0{,}c_1c_2...c_{cc}ef..\ 10^{\alpha}$, alors le nombre de chiffres significatifs cs de leur différence sera: cs = cmax-cc et $a-b = 0{,}c'_1c'_2...c'_{cs}..\ 10^{\alpha - cc}$.

On obtient ainsi des pertes des propriétés mathématiques : L'associativité par exemple n'est plus vraie.

$(x+y) + z \neq x + (y+z)$

Ex: si on se place en base 10 avec une précision de 10 chiffres, et $x = -1{,}234567809.10^{16}$, $y = 1{,}234567809.10^{16}$ et $z = 1000$, on obtient :

$(x+y) + z = 1000$

$x + (y+z) = 0$

L'addition est donc non associative par suite d'erreur d'arrondi.

De même, l'ordre des opérations de

### 3.3. Erreur sur une addition arithmétique

Soit + l'addition exacte et $\underline{+}$ l'addition informatique. Soient $x_1$ et $x_2$ deux réels représentés dans **F** par $X_1$ et $X_2$, alors

$$X_i = x_i - 2^{E_i - t}\delta_i\alpha_i \qquad \text{pour } i = 1, 2$$

Pour l'addition de $X_1$ et $X_2$ nous obtenons :

$$X_1 \underline{+} X_2 = X_1 + X_2 - 2^{E_3 - t}\delta_3\alpha_3$$

$$X_1 \underline{+} X_2 = x_1 + x_2 - 2^{E_1 - t}\delta_1\alpha_1 - 2^{E_2 - t}\delta_2\alpha_2 - 2^{E_3 - t}\delta_3\alpha_3$$

Les valeurs $E_3$, $\delta_3$, $\alpha_3$ dépendent de $\alpha_1$ et $\alpha_2$.

### 3.4. Erreur sur une soustraction informatique

Soit – la soustraction exacte et $\underline{-}$ la soustraction informatique.

Pour la soustraction de $X_1$ et $X_2$ nous obtenons :

$$X_1 \underline{-} X_2 = X_1 - X_2 - 2^{E_3 - t}\delta_3\alpha_3$$

$$X_1 \underline{-} X_2 = x_1 - x_2 - 2^{E_1 - t}\delta_1\alpha_1 - 2^{E_2 - t}\delta_2\alpha_2 - 2^{E_3 - t}\delta_3\alpha_3$$

Pour l'addition et la soustraction, on n'a eu recours à aucune hypothèse ou approximation; il n'en sera pas de même pour la multiplication et la division.

### 3.5. Erreur sur une multiplication informatique

Soit $\times$ la multiplication exacte et $\underline{\times}$ la multiplication informatique.

Pour la multiplication :

$$X_1 \underline{\times} X_2 = X_1 \times X_2 - 2^{E_3 - t}\delta_3\alpha_3$$

$$X_1 \times X_2 = x_1 \times x_2 - x_2.2^{E_1 - t}\delta_1\alpha_1 - x_1.2^{E_2 - t}\delta_2\alpha_2 + 2^{E_1 + E_2 - 2.t}\delta_1\delta_2\alpha_1\alpha_2$$

En négligeant le terme du deuxième ordre, on obtient :

$$X_1 \underline{\times} X_2 = x_1 \times x_2 - x_2.2^{E_1 - t}\delta_1\alpha_1 - x_1.2^{E_2 - t}\delta_2\alpha_2 - 2^{E_3 - t}\delta_3\alpha_3 + O(2^{-2t})$$

### 3.6. Erreur sur une division informatique

Soit / la multiplication exacte et $\doteq$ la division informatique.

De la même façon, pour la division :

$$X_1 \doteq X_2 = X_1/X_2 - 2^{E_3-t}\delta_3\alpha_3$$

$$X_1/X_2 = (x_1 - 2^{E_1-t}\delta_1\alpha_1).(x_2 - 2^{E_2-t}\delta_2\alpha_2)^{-1}$$

En négligeant les terme du deuxième ordre en $2^{-t}$, on obtient :

$$X_1 \doteq X_2 = x_1/x_2 - 2^{E_1-t}\delta_1\alpha_1/x_2 + 2^{E_2-t}\delta_2\alpha_2 x_1/x_2^2 - 2^{E_3-t}\delta_3\alpha_3 + O(2^{-2t})$$

### 3.7. Programmation avertie

On peut éviter certains problèmes de cancellation (anulation de digits) en organisant les calculs de manières différentes. Nous présentons ici quelques exemples.

- Résolution d'une équation du second degré $ax^2 + bx + c = 0$

Supposons que a est non nul et que le discriminant est positif. Le calcul des racines s'effectue au moyen de la formule :

$$x = \frac{-b \pm \sqrt{b^2 - 4.a\,c}}{2.a}$$

Si $b^2$ est très grand par rapport à 4.a c, alors les valeurs de b est du discriminant seront très proches (en valeurs absolues), et au cours de la soustraction de ces deux valeurs, on a une grande perte de chiffres significatifs et une des racines de l'équation est complètement fausse.

Exemple : $a = 10^{-4}$, b= -0,8, $c = 10^{-4}$, sur un ordinateur à 6 chiffres sinificatifs en base 10, on obtient : $x_1 = 8000$ et $x_2 = 0$.

Or les valeurs exactes sont : $x_1 = 7999.999875$ et $x_2 = 0.000125$.

La valeur de la deuxième racine est complètement faussée car l'algorithme a soustrait deux valeurs très proches.

Pour éviter cette erreur, il suffit de ne pas effectuer la soustraction des deux nombres proches. Pour cela, on utilise les relations suivantes :

$$\begin{cases} x_1 = \dfrac{-b + \varepsilon\sqrt{b^2 - 4.a\,c}}{2.a} \\ x_2 = \dfrac{c}{a * x_1} \end{cases} \qquad \text{avec } \varepsilon = -\text{signe}(b)$$

On obtient alors une valeur correcte pour $x_2$.

• De même, lors du calcul de $\frac{1}{x} - \frac{1}{x+1}$, si x est grand, il est préférable d'utiliser la formule $\frac{1}{x(x+1)}$ qui est beaucoup plus stable.

On peut encore trouver d'autres exemples de formulations algébriques qui permettent d'éviter certaines soustractions catastrophiques [BRE 88] [ GOL 91][VIG 93].

## 4. APPROCHES DE SOLUTIONS AU PROBLEME DE L'IMPRECISION NUMERIQUE

### 4.1. Outils mathématiques

On peut étudier l'influence des erreurs de données sur les résultats.

#### *4.1.1. Analyse différentielle*

Pour étudier la propagation des erreurs au cours des calculs, on peut utiliser l'analyse différentielle (développement de Taylor) [STO 93].

Soit $f : \Re^n \longrightarrow \Re^m$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = f(x) = \begin{bmatrix} f_1(x_1,\ldots,x_n) \\ \vdots \\ f_m(x_1,\ldots,x_n) \end{bmatrix}$$

alors les erreurs absolues $e_{xi}$ des $x_i$ ( i = 1,...,n) entraînent des erreurs absolues $e_{yj}$ sur les résultats $y_j$ ( j=1,...,m) qui sont données par la formule :

$$e_{yi} \approx \sum_{j=1}^{n} \frac{\partial f_i(x)}{\partial x_j} e_{xj} \quad (i = 1,\ldots,m).$$

Si on considère la même fonction que précédemment, alors les erreurs relatives $er_{xi}$ des $x_i$ (i = 1,...,n) entraînent des erreurs relatives $er_{yj}$ sur les résultats $y_j$ ( j=1,...,m) qu'on peut estimer à l'aide de la formule :

$$er_{yi} \approx \sum_{j=1}^{n} \frac{x_j}{f_i(x)} \frac{\partial f_i(x)}{\partial x_j} er_{xj} \quad (i = 1,\ldots,m).$$

Cet outil est utilisé, au chapitre II paragraphe 3, pour le cas particulier du calcul du point d'intersection de deux droites.

*4.1.2. Conditionnement d'un problème*

Savoir si un problème est bien ou mal conditionné, c'est étudier comment les perturbations sur les données d'un problème influent sur les résultats. On dit qu'un problème est bien (mal) conditionné si une petite variation des données entraîne une petite (grande) variation sur les résultats.

Cette notion de conditionnement peut-être introduite de la façon suivante:

Soit P(a) un problème dont les données sont représentées par le vecteur a, de composantes $a_1, a_2,\ldots,a_m$, et dont les résultats sont représentés par le vecteur x de composantes $x_1, x_2,\ldots,x_n$. Si f est l'application de $\Re^m$ dans $\Re^n$, qui à a associe x ( $x = f(a)$ ), et si $\delta a$ est une variation de la donnée a, alors $\delta a$ produit une variation $\delta x$ des résultats et l'on a : $x + \delta x = f(a + \delta a)$. Si selon le problème, on peut trouver une constante M telle que $\|\delta x\| \leq M \times \|\delta a\|$ pour $\|\delta a\| \leq \varepsilon$ donné, alors le problème P(a) sera bien (mal) conditionné si la constante M est "petite" ("grande").

On peut tout aussi bien étudier le conditionnement des racines d'un polynôme vis-à-vis des variations de ses coefficients, que celui des valeurs propres (ou vecteurs propres) d'une matrice par rapport aux variations de ses éléments, ou, comme le fait M. Daniel [DAN 89], étudie le conditionnement d'un changement de base qui lui permet d'estimer les ordres de grandeur des perturbations relatives des coefficients des polynômes dans les deux bases. Il montre ainsi que les matrices de passage de la base de Bernstein à la base canonique sont mal conditionnées et que les risques de perte de chiffres significatifs sur les coefficients des polynômes sont importants.

C. M. Hoffmann [HOF 89] recherche aussi la précision avec laquelle une matrice peut être inversée pour le calcul du point d'intersection de deux droites. Si $\beta$ est l'angle entre les deux droites, alors en supposant $\beta < 180°$, on peut montrer que $\mathrm{Cond}(A) = \frac{4}{\sin(\beta)}$.

Cette estimation montre que le problème est bien conditionné pour des angles proches de 90° et est mal conditionné pour des angles proches de 0° et 180°. Sous des circonstances favorables, on peut s'attendre à une perturbation des coefficients de l'ordre de $2^{-t}$, dû à un arrondi du dernier digit représentable (t est la longueur de la mantisse). Mais pour des angles petits, on a $\sin(\beta) \approx \beta$ et pour un angle de $1/2^m$, on peut s'attendre à perdre $2m + 2$ digits binaires.

M. La Porte et J. Vignes [VIG 74] ont étudié le moyen de vérifier si une relation, qui doit être satisfaite, l'est effectivement. Ils ont proposé une méthode de contrôle de la solution, d'un système linéaire, d'une équation algébrique, et le contrôle du résultat d'un algorithme d'inversion de matrice.

Le principe pour un système linéaire est le suivant (les deux autres problèmes étant résolus de façon similaire): Considérons un système linéaire non dégénéré :

$$A \times X - B = 0$$

avec $A = [a_{ij}]$, $B = [b_i]$, $1 \leq i,j \leq N$. Si $X$ est la solution exacte obtenue par voix algébrique ou par voie numérique exacte, la méthode la plus simple pour valider la solution, est de remplacer, dans chaque équation du système, les inconnues par les valeurs trouvées et de regarder si tous les termes s'éliminent. Si c'est le cas, alors la solution est satisfaisante, sinon elle est incorrecte.

Sur le plan informatique, nous avons des représentations A et B en machine de $A$ et $B$, et le système à résoudre devient :

$$A \times X - B = 0$$

Pour contrôler la validité de la solution obtenue X, on est conduit à faire calculer par l'ordinateur le résidu R défini par :

$$R = (A \times X - B)_{machine}$$

Même si X est la solution exacte, le résidu R sera en général non nul, à cause de la précision finie de la machine. Si ce résidu ne dépasse pas une certaine tolérance, alors la solution peut être considérée exacte, sinon non.

M. La Porte et J. Vignes proposent de déterminer de façon théorique une estimation $\hat{R}_i$ du résidu $R_i$, et un nouveau test d'arrêt qui ne dépend plus d'une tolérance quelconque. Ils obtiennent les expressions suivantes :

- pour la troncature :

$$\hat{R}_i = 2^{-n} N \sqrt{\left\{ \sum_{j=1}^{N} \left(a_{ij} x_i\right)^2 \right\} + b_i^2}$$

- pour l'arrondi :

$$\hat{R}_i = 2^{-n} \sqrt{N \left\{ \left\{ \sum_{j=1}^{N} \left(a_{ij} x_i\right)^2 \right\} + b_i^2 \right\}}$$

avec n le nombre de bits de la mantisse. Ensuite, pour chaque équation, ils calculent le résidu normé $\rho_i$:

$$\rho_i = \frac{|R_i|}{\hat{R}_i}$$

Trois cas peuvent être envisagés:

*1^er cas*. Les N résidus normés sont de l'ordre de grandeur de l'unité, $\rho_1, \rho_2, \ldots, \rho_N \approx 1$, alors la solution est satisfaisante.

*2ème cas*. L'un au moins des résidus normés est nettement supérieur à l'unité, mais nettement inférieur à $2^n$, $1 << \rho_i << 2^n$, alors la solution n'est pas pleinement satisfaisante. En général, elle peut être améliorée sans utilisation de la double précision. Il suffit d'appliquer à nouveau l'algorithme de résolution en remplaçant B par B - A×X.

*3ème cas.* L'un au moins des résidus est de l'ordre de grandeur de $2^n$, $\rho_i \approx 2^n$, alors la solution est mauvaise, elle ne peut généralement pas être améliorée. Cela signifie que la méthode de résolution est mal adaptée au système proposé.

Cette solution semble coûteuse en calcul et donc en temps de résolution, et de plus les tests qui concluent si on a une bonne ou mauvaise solution sont aussi des estimations, et cela n'apporte rien en ce qui concerne la précision des calculs.

### *4.1.3. Permutation-Perturbation*

La méthode *Permutation-Perturbation* est présentée succinctement par D. Michelucci [MIC 87] et par G. Masotti [MAS 93]. Elle a été introduite par J. Vignes et M. La Porte [VIG 78], [VIG 87] et détaillée dans [VIG 93]. Elle peut estimer le nombre de digits significatifs dans le résultat de l'algorithme. La valeur du coût total de l'application de la méthode pour l'estimation du nombre de digits significatifs, avec un niveau de confiance à 0.95 (intervalle de confiance d'une probabilité de 0.95, cf chapitre III), semble introduire un facteur de pénalité plus petit que 3 dans le temps total d'exécution.

Elle consiste à calculer une valeur de plusieurs façons différentes, mathématiquement équivalentes, par *permutation* des termes, et par *perturbation* aléatoire du dernier bit. Cela donne un échantillon de valeurs qui permet d'obtenir des encadrements fiables $\left[u_0^*, u_1^*\right]$ pour les valeurs calculées de u.

Cette méthode permet donc de connaître la précision d'un calcul, et donc de prévenir qu'il y a un mauvais déroulement numérique du programme, mais elle n'évite pas, ni ne diminue les erreurs. On a inévitablement une augmentation du temps de calcul et l'ordre exact entre deux nombres n'est pas plus fiable.

Une étude plus approfondie de la méthode et une application à l'intersection de deux droites sont développées au chapitre III.

### *4.1.4. La différentiation automatique*

La différentiation automatique [EYS 96] est une technique permettant d'obtenir des dérivées exactes (aux erreurs d'arrondi près) d'une fonction représentée par une programme informatique. Si c'est un programme qui représente la fonction à dériver, c'est un autre programme qui est généré par le différentiateur. Il faut alors faire tourner ce dernier pour obtenir la valeur de la fonction en un point donné.

Un des attraits de la différentiation automatique est de pouvoir générer un programme calculant le gradient d'une fonction à valeurs scalaires, dont le temps d'exécution relatif à celui du programme original est indépendant du nombre de variables par rapport auxquelles on dérive. Il s'agit en fait de génération automatique de codes adjoints (codes linéaires cotangents).

De plus, en double précision, on peut avoir entre 1 et 3 chiffres significatifs corrects de plus que par des différences finies usuelles.

## 4.2. Techniques de résolution

### *4.2.1. Utilisation d'une tolérance*

Une idée classique pour résoudre le problème des incohérences numériques, est l'emploi d'une précision qui représente la tolérance, souvent notée ε, admise pour déterminer une égalité. Par exemple, si deux nombres a et b diffèrent de moins de ε, alors a et b sont considérés comme égaux.

Le choix d'un epsilon correct est très délicat : la valeur de l'epsilon est dépendante de l'application et de l'ordre de grandeur des nombres les plus fréquemment utilisés. Cette technique améliore les résultats des algorithmes, mais ne résout que partiellement certains problèmes. En effet, il existe plusieurs façons mathématiquement équivalentes de calculer deux nombres a et b; il se peut que pour un premier mode de calcul a < b à un epsilon près, et que pour un deuxième a > b pour ce même epsilon. Ces deux résultats ne peuvent être tous les deux corrects; comment décider alors de façon sûre l'ordre de a et b ?

La meilleure approche pour réduire les effets de l'imprécision est d'augmenter la précision, c'est à dire le format des réels. Mais il n'y a toujours pas de contrôle sur la génération et la propagation d'erreur.

G. Masotti [MAS 93] propose une méthode pour obtenir une *estimation précise* de l'erreur absolue qui affecte les résultats.
Elle consiste à calculer tout au long de l'algorithme l'erreur absolue de chaque réel. Pour cela il représente un nombre réel par sa valeur x, l'estimation de l'erreur qui lui est associée ee et l'erreur relative maximale re. Comme l'accumulation d'erreur est graduelle, il se peut qu'à un certain moment de l'algorithme, re dépasse un seuil d'acceptabilité dépendant de la précision utilisée pour représenter les nombres en virgules flottantes, et l'utilisateur pourra en être prévenu. Le réel est donc représenté par le triplet :

**(x,ee,re)**

Il détermine la contribution de l'erreur durant le calcul pe à l'aide d'un développement de Taylor, décrit au paragraphe 4.1.1.

A cette erreur, il ajoute l'erreur d'arrondi qu'effectue l'ordinateur. Pour la déterminer, il suppose qu'à un moment il a, à sa disposition, le résultat w avec un nombre de bits significatifs plus grand que celui du format du réel, et dans ce cas on a :

z = rd(w) fonction d'arrondi qui opère sur w pour obtenir z dans le format déclaré
le = z - w erreur de calcul local

Le résultat d'un calcul est donc le triplet $(z, e_z, re_z)$, où $e_z = rd(pe + le)$ et $re_z = |e_z/z|$.

L'objectif de G. Masotti [MAS 93] est d'utiliser cette estimation de l'erreur pour déterminer un intervalle de confiance pour la vraie erreur, et donc pour la vraie valeur du réel considéré. Ce qui lui permet de comparer deux nombres avec plus de précision : si a et b sont les deux nombres à comparer, il construit les deux intervalles de confiance et il vérifie s'ils se chevauchent. S'ils sont disjoints, alors a et b ne représentent pas le même nombre, sinon ils sont considérés comme égaux.

Cette méthode permet donc de contrôler les erreurs accumulées et d'en tenir compte pour les comparaisons de réels. Elle introduit néanmoins quelques problèmes :

- La place mémoire demandée pour représenter un réel augmente sensiblement, puisque, en plus du nombre, il stocke l'erreur absolue et l'erreur relative. Il ne parait pas intéressant de conserver cette dernière valeur, car elle n'intervient pas dans la suite des calculs; il suffit de tester l'erreur relative par rapport au seuil d'acceptabilité juste après le calcul.
- Le temps de calcul augmente aussi puisqu'il faut, pour chaque opération, évaluer l'erreur. Il propose de diminuer ce temps en utilisant des machines parallèles. Cela permettrait d'effectuer plusieurs calculs en même temps, mais l'accumulation des opérations au cours des calculs introduit quand même une augmentation non négligeable du temps d'exécution puisque la détermination d'une erreur est effectuée à chaque opération d'un calcul. Par exemple, le calcul de 3*x + 4*x*x - 5 nécessite 5 opérations élémentaires plus une dizaine d'opérations supplémentaires, sans compter la détermination de l'erreur d'arrondi.

De plus, pour comparer deux nombres, il faut calculer des intervalles et les comparer, ce qui prend aussi du temps.

### *4.2.2. Arithmétique rationnelle*

a) Introduction

La méthode la plus efficace pour éliminer les erreurs numériques est l'utilisation de nombres entiers. Dans ce cas, on ne travaille plus qu'avec des entiers ou des rationnels. Plusieurs chercheurs utilisent cette arithmétique pour améliorer la précision de leur modèles.

b) Application à l'intersection de deux droites

i) Dominique Michelucci

Dans sa thèse, D. Michelucci [MIC 87] propose d'utiliser l'arithmétique rationnelle pour la construction d'une BREP (Boundary REPresentation), dite carte planaire, qui décrit les scènes 2D polygonales afin d'éviter les méfaits de l'imprécision numérique, qu'il met en évidence surtout en ce qui concerne la comparaison de nombres.

Il suppose que les sommets initiaux de la carte sont recalés sur une très grande grille carrée entière $[0..G^2]$, où G est déterminé suivant les besoins et la machine utilisée.

Pour construire une carte planaire, il procède en trois étapes :

- la construction d'une structure de données locale (CP-locale) qui contient les informations relatives à chaque sommet initialement connu et aux arêtes incidentes.
- la détermination des points d'intersection entre chaque segment et la mise à jour au fur et à mesure de la CP-locale : addition des sommets, divisions des segments qui se coupent (algorithme de Bentley-Ottmann en annexe 1).
- la construction de la CP-globale à partir de la CP-locale, qui indique explicitement pour chaque arête à quels contours elle appartient et les arêtes adjacentes, et inclusions entre les divers contours de la carte planaire.

Cependant, l'algorithme de Bentley-Ottmann, détaillé en annexe 1, compare des réels, et exploite les propriétés des relations d'ordre, comme la transitivité, pour l'y-ordre sur les arêtes et pour l'xy-ordre sur les coordonnées ( si les informations $a < b$ et $b < c$ ont été établies, alors un algorithme efficace en déduit sans calcul supplémentaire que $a < c$). Malheureusement, les comparaisons sont le résultat de calculs effectués en arithmétique flottante, d'où le risque important d'erreurs au cours de l'exécution de cet algorithme.

Afin d'améliorer son efficacité, D. Michelucci [MIC 87] utilise une arithmétique rationnelle pour calculer avec précision l'intersection de deux droites :
La droite $( (x_0,y_0);(x_1,y_1) )$ a alors pour équation $ax + by + c = 0$ avec $a = y_0-y_1$, $b = x_1-x_0$, $c = x_0y_1 - x_1y_0$. Les nombres $|a|$, $|b| \leq G$ et $|c| \leq 2G^2$ sont entiers, et le point d'intersection M entre deux droites est alors rationnel. Si on a le système $\begin{cases} ax+by+c=0 \\ a'x+b'y+c'=0 \end{cases}$
On a alors des déterminants $\Delta$, $\Delta x$, $\Delta y$ entiers, et M a pour coordonnées $\Delta x/\Delta$ et $\Delta y/\Delta$.

Remarque :

Pour représenter un tel point en machine, il préfère utiliser la représentation euclidienne ($x_e$, $y_e$, $x_r$, $y_r$, $\Delta$ avec $\Delta x = x_e\Delta + x_r$, et $\Delta y = y_e\Delta + y_r$), plutôt que la représentation sous forme de coordonnées homogènes entières ($\Delta x$, $\Delta y$, $\Delta$), pour laquelle la valeur de G doit être plus grande, car les valeurs calculées avec la première représentation ( $0 \leq x_e, y_e \leq G$, $0 \leq x_r, y_r \leq 2G^2$) seront plus petites qu'avec la deuxième ($|\Delta x|$ et $|\Delta y| \leq 4G^3$, et $|\Delta| \leq 2G^2$). Cependant, cette représentation nécessite de conserver cinq valeurs par point et une gestion particulière pour effectuer les calculs.

Avec cette représentation, la comparaison des nombres rationnels devient exacte en faisant appel au développement en fractions continues :ordre(a/b;c/d) = ordre(d/c;b/a) = ordre([d/c] +

(d%c)/c;[b/a] + (b%a)/a), avec [b/a] est la partie entière de b/a, et b%a le reste de cette division. On obtient la récursivité suivante:
si (a/b) et (c/d) sont entiers
alors on a une comparaison normale d'entiers
sinon si [d/c] ≠ [b/a]
alors ordre(a/b;c/d) = ordre([d/c];[b/a])
sinon ordre(a/b;c/d) = ordre((d%c)/c;(b%a)/a).

Cette méthode permet à l'algorithme de Bentley-Ottmann (Annexe 1) de fonctionner correctement.

ii) Christopher Hoffmann

Ce procédé est également présenté par C. Hoffmann [HOF 89]. Il considère des polyèdres dans lesquels les faces sont données numériquement et toutes les autres informations sont symboliques ( les sommets sont définis comme l'intersection des trois plans distincts, les droites portants les arêtes comme l'intersection de deux plans distincts).
L'équation d'un plan est de la forme ax+by+cz+d = 0 où a, b, c, d sont des entiers bornés, $|a|$, $|b|$, $|c| \leq L$ et $|d| \leq L^2$, où L est défini de manière à éviter le dépassement de la précision de la machine lors des calculs.
Il impose qu'un polyèdre soit construit à partir d'une suite d'opérations sur des primitives qui doivent être triviales et satisfaire un minimum de conditions : chaque arête est plus longue qu'un minimum défini auparavant, l'angle entre deux faces adjacentes ne peut pas être plus petit qu'un angle minimal , et il exige qu'un polyèdre trivial soit tel que chaque sommet est incident à trois faces.

Ce type de représentation engendre beaucoup de difficultés et d'inconvénients :
- Limitation des nombres représentables
- Perte de précision et de caractéristiques au cours des transformations
- Introduction volontaire d'erreurs

• Limitation des nombres représentables :

L'utilisation d'une grille rationnelle restreint l'utilisation de grands nombres, par exemple, pour une machine à 32 bits où le plus grand entier représentable est $2^{31}$-1, il faudrait que $2G^2 < 2^{31}$, soit G < 23768, et s'il devient nécessaire d'employer des valeurs excédant les possibilités de la machine, on devrait être obligé d'utiliser des flottants double précision pour représenter les entiers dans les limites de la mantisse ou utiliser une librairie arithmétique, manipulant des entiers de longueur quelconque, qui ralentirait incontestablement l'exécution de l'algorithme.

De même, comme le constate D. Michelucci [MIC 87], si on représente des plans sous cette forme, les entiers stockés peuvent atteindre $144G^7$. Par exemple, pour obtenir une précision du millimètre dans un univers d'un kilomètre cube, (applications architecturales et urbanistiques), il faut une valeur de $G = 10^6$, et les entiers à stocker peuvent atteindre $48G^6 = 48.10^{36}$ ($\approx 2^{125\ 174...}$), ou même $144G^7 = 144.10^{42}$ ($\approx 2^{146\ 690...}$), une arithmétique est alors nécessaire pour les représenter fidèlement en machine, et le coût de cette précision devient très important.

• Perte de précision et de caractéristiques au cours des transformations :

Si l'on effectue une translation ou une rotation, il est clair que le résultat n'est pas représentable. Par exemple, le plan résultant P' d'une rotation d'un plan P n'aura pas forcément des coefficients entiers. Les opérations seront donc exécutées approximativement sans pour autant violer l'intégrité des objets en *arrondissant les éléments* de façon à avoir un plan P" représentable proche de P'. Cela nécessite deux types d'approximations :

1. Soit un réel positif w, trouver un rationnel p/q approchant w tel que q n'excède pas une borne Q.

2. Soit l'équation d'un plan $ax + by + cz + d = 0$, trouver une équation représentable approchant le plan, c'est à dire trouver un plan $a'x + b'y + c'z + d' = 0$ avec des coefficients entiers tel que $|a'|, |b'|, |c'| \leq L$ et $|d'| \leq L^2$.

Ces approximations sont présentées par C. Hoffmann [HOF 89], mais elles ne garantissent pas que toutes les caractéristiques d'un objet seront conservées. Par exemple, un polygone peut être transformé en un polygone dont les arêtes se coupent.

• Introduction volontaire d'erreurs :

L'approximation d'un réel par un rationnel (par exemple, après une transformation) est effectuée à une précision près. Il faudrait tenir compte de cette erreur dans la suite, mais D. Michelucci n'en parle pas.

De plus, la représentation d'une droite par des entiers se rapproche d'une représentation en coordonnées homogènes ( (x, y, t) représente le point de coordonnées (x/t, y/t) ), par exemple, le point de coordonnées (0,15; 1,154) peut être considéré sous la forme (150; 1154; 1000), que l'on peut encore restreindre en calculant le pgcd des trois entiers (75; 577; 500). Mais on a toujours une restriction des entiers.

D'ailleurs, l'arithmétique rationnelle (supposée exacte) est très coûteuse, aussi bien en temps de calculs qu'en place mémoire.

c) Optimisations

Diverses optimisations ont été proposées comme celle de J.M Moreau [MOR 90] (arithmétique exacte réticente), qui consiste à utiliser au mieux les capacités de l'arithmétique

flottante et à ne faire appel au calcul rationnel qu'en dessous de certains seuils calculés à l'avance.

Dans le même esprit, M.O. Benouamer [BEN 93], présente une arithmétique rationnelle dite paresseuse. Elle permet d'effectuer les calculs exacts strictement nécessaires. Chaque nombre paresseux x est représenté par un intervalle flottant contenant la valeur exacte de x et par une "définition" qui permet de calculer au besoin cette valeur à l'aide de nombres entiers et rationnels, ce qui permet une bonne approximation. Une "définition" peut être soit un rationnel, soit une expression symbolique représentant x comme la somme, le produit, l'opposé ou l'inverse d'autres nombres paresseux. Elle consiste en fait en un arbre dont les noeuds internes désignent des opérateurs arithmétiques unaires ou binaires et dont les feuilles désignent des nombres rationnels ordinaires.
Bien que cette méthode augmente la place mémoire pour représenter un nombre et nécessite une arithmétique spéciale, elle s'avère avantageuse pour son efficacité.

*4.2.3 Arithmétique d'intervalle*

Un intervalle réel fermé I est de la forme : $I = [a, b]$ avec $a \leq b$ et $a, b \in \mathbf{R}$. Soit $J$ l'ensemble des intervalles réels fermés. Sur les éléments de $J$, on peut introduit une arithmétique définie algébriquement comme suit :

- $[A, B] + [C, D] = [A+C, B+D]$
- $[A, B] - [C, D] = [A-D, B-C]$
- $[A, B] \times [C, D] = [\min(AC, AD, BC, BD), \max(AC, AD, BC, BD)]$
- $1 \div [A, B] = \begin{cases} [1/B, 1/A] & \text{si } A.B > 0 \\ [1/B, +\infty] & \text{si } A = 0 \\ [-\infty, 1/A] & \text{si } B = 0 \\ [-\infty, +\infty] & \text{si } A < 0 < B \end{cases}$
- $[A, B] \div [C, D] = [A, B] \times (1 \div [C, D])$.

Remarque : Sur un ordinateur, il serait plus exact, pour mieux contrôler les erreurs d'arrondi durant les calculs, de prendre l'arrondi par défaut pour la borne inférieure de l'intervalle et l'arrondi par excès pour la borne supérieure, mais cela est difficile à gérer durant l'exécution d'un programme, puisque l'ordinateur effectue lui-même les arrondis.

Le fait que les valeurs numériques exactes ou provenant de calculs intermédiaires ne soient pas connues, incite à considérer des petits intervalles contenant ces valeurs et bornés par des nombres appartenant à l'ensemble **F** des réels représentables en machine [MUL 89].

Soit x un réel que l'on veut représenter par un intervalle, ce dernier peut être défini de différentes manières :

- $[\nabla(x), \Delta(x)]$ où $\nabla$ est l'arrondi par défaut : $\nabla(x) := \max\{y \in R \mid y \leq x\}$

  $\Delta$ est l'arrondi par excès : $\Delta(x) := \min\{y \in R \mid y \geq x\}$

- $[x - \varepsilon, x + \varepsilon]$ où $\varepsilon$ est une précision absolue à préciser suivant la grandeur de x et les cas particuliers qui peuvent intervenir.

- $[r1(x), r2(x)]$ où $r1(x)$ et $r2(x)$ sont des bornes rationnelles encadrant x et telles que la grandeur de l'intervalle soit plus petite qu'une certaine tolérance. Le logiciel de calcul formel Maple utilise ce procédé pour encadrer les résultats de certaines résolutions mahtématiques.

- il existe d'autre types d'arrondi (arrondi au plus près, vers zéro, vers l'infini, stochastique) [Mul 89], qui peuvent permettre de définir d'autres intervalles.

Cette arithmétique est appliquée à l'intersection de deux droites et détaillée au chapitre II paragraphe 4 de ce mémoire.

### *4.2.4. Représentation symbolique*

•*Représentation et modèle*

Une *représentation* est une structure de données destinée à décrire un objet géométrique, avec la possibilité d'utiliser des données arithmétiques imprécises. Elle comprend des données symboliques décrivant les états adjacents et incidents, et des données numériques telles que les coefficients de chaque face. Une telle représentation a un *modèle* s'il existe un objet dans l'espace euclidien satisfaisant la partie symbolique de la description avec précision. Aux données numériques de la représentation, il correspond des données numériques du modèle qui peuvent nécessiter des nombres en précision infinie [HOF 89].

Pour avoir une représentation qui corresponde bien à un modèle, C. Hoffmann [HOF 89] va exiger que les données numériques de la représentation soient proches des données du modèle. On dira qu'un modèle $M$ d'une représentation $R$ est à $\varepsilon$-près, si l'écart entre les données numériques de la représentation et celles du modèle n'est pas plus grand que $\varepsilon$. C'est une notion d'erreur absolue, mais elle peut être remplacée par une définition d'erreur relative.

Avec ces concepts, il clarifie quand une opération *op* d'ordre k est correctement implantée: la mise en oeuvre sera correcte, si, pour toute représentation donnée $R_i$, il existe un modèle $M_i$ tel que :

1. L'algorithme construit une représentation R sans défaut.
2. Il y a un modèle $M$ de $R$ tel que $M = op\ (M_1,..,M_k)$

Il affine la définition en imposant que, pour chaque modèle $M_i$ à $\varepsilon$ près de sa représentation, le modèle $M$ est à $\delta(\varepsilon)$ près de $R$, où $\delta$ est une fonction telle que $\delta(\varepsilon)$ ne soit pas trop grand par rapport à $\varepsilon$.

•*Représentation purement symbolique*

Ce type de représentation a pour objet d'éliminer le rôle du calcul numérique et est étudié par C. Hoffmann [HOF 89]. Comme exemple, il considère des objets géométriques constitués de droites, données par [a, b, c], et de points, donnés par (u, v, w), où a, b, c, u, v et w sont des symboles. Il se place dans l'espace projeté de dimension 2. Le triplet [a, b, c] symbolise l'équation de la droite ax + by + cz = 0, où z est la variable homogénéisée, le triplet (u, v, w) sont les coordonnées du point projeté, correspondant au point affine (u/w, v/w) quand $w \neq 0$. Dire que le point P = (u, v, w) est *incident* à la droite L = [a, b, c] signifiera que l'équation au + bv + cw = 0 pourra être satisfaite et ce fait sera noté L(P).
Il détermine un arrangement de points et de droites par les règles suivantes :

(D1) Toutes droites et points doivent être déclarées à l'avance comme des triplets de symboles. On ne peut pas avoir deux droites ou deux points ainsi déclarés égaux.

(D2) Si un point P est incident à la droite L, alors ce fait est déclaré comme L(P). Si deux droites L1 et L2 se coupent au point P déclaré, alors ce fait est exprimé par les deux états d'incidences L1(P) et L2(P).

(D3) Aucune autre incidence entre les points et droites déclarés existe, mis à part celles déclarées explicitement.

Avec cette représentation qui omet les valeurs numériques, C. Hoffmann montre facilement à l'aide d'un exemple qu'il y a un risque de perte de l'existence des objets. Pour pouvoir valider cette existence, il faudrait, après chaque opération géométrique de base sur les représentations, vérifier s'il existe un modèle pour chaque objet. De tels algorithmes nécessiteraient des calculs symboliques de capacité excessive.

•*La règle de raisonnement*

Dans les algorithmes, on est souvent amené à tester si x est plus petit, égal, ou plus grand qu'une certaine valeur. C'est un point délicat qui peut entraîner des défauts, car ces tests sont effectués suivant une précision définie par une constante $\varepsilon$. Les risques d'avoir de fausses conclusions augmentent lorsque les valeurs proviennent de calculs intermédiaires. En effet, si les erreurs (accumulées au cours des calculs) sur un nombre x sont plus grandes que la tolérance $\varepsilon$, alors quand on compare x avec zéro, la réponse sera négative ($x \neq 0$), même si x devrait être nul théoriquement.

Pour essayer de réduire ces risques, il faudrait prendre de telles décisions en fonction des données topologiques. C. Hoffmann [HOF 88] et [HHK 89] s'est basé sur cette idée pour développer des algorithmes pour lesquels les opérations sur des polyèdres sont robustes. Il présente une règle de raisonnement qui considère les informations numériques comme étant des approximations des données réelles et cherche à trouver les informations à partir des

données symboliques décrivant la topologie de l'objet. De manière plus générale, il utilise le résultat du calcul en nombre flottant quand il est sans danger, sinon il prend les décisions suivant les données topologiques.

Cette méthode semble satisfaisante pour les polyèdres, car elle permet de réduire les intervalles critiques pour les données. Par exemple, une rotation pourra être effectuée avec un angle plus petit, sans que l'objet final et l'objet initial soient confondus. Cependant, cette méthode ne permet pas de maîtriser les ε, d'ailleurs C. Hoffmann se fixe un epsilon incertain, indépendant des calculs précédemment effectués.

*4.2.5. Modification des données symboliques*

V. J. Milenkovic [MIL 88] propose deux méthodes pour la création d'algorithmes géométriques corrects en précision finie : la *normalisation des données* et la méthode de *la variable cachée*.

•*Normalisation des données*

Le but de cette méthode est de modifier la structure et les paramètres d'un objet géométrique pour obtenir un objet pour lequel les tests numériques seront exacts. Il applique cette méthode pour la modélisation de régions polygonales dans le plan. Pour définir ce système, il se donne un ε tel que la distance entre deux points et entre un point et une droite, puisse être calculée avec une précision de $\frac{1}{10}\varepsilon$; et cinq règles pour la normalisation :

1) Pas deux sommets distincts proches à moins de ε.
2) Pas de sommet proche à moins de ε d'une arête s'il n'appartient pas à l'arête
3) Pas deux arêtes qui se coupent excepté en leur points communs.
4) Pour chaque sommet, la liste triée des arêtes arrivant et partant de ce sommet.
5) Pour chaque point du plan, le nombre topologique onduleux ("winding") (0, 1 ou indéfini) qui indique s'il est à l'extérieur, l'intérieur ou sur le bord d'un polygone.

Durant des opérations de base, on peut violer les règles 1) et 2) et on doit alors modifier les polygones pour qu'ils respectent les règles. Deux opérations de bases sont nécessaires : *déplacement de sommets* ("*vertex shifting*") et *éclatement d'arêtes* ("*edge cracking*").

Déplacement de sommets : Si un point U est proche à moins de ε d'un sommet V, alors on déplace V à l'emplacement de U et on élimine les doubles arêtes (Figure 3).


Figure 3 : Déplacement de sommets

Éclatement d'arêtes : Une arête qui passe à ε près d'un sommet viole la règle 2) et est éliminée de la façon suivante (Figure 4) :

Étape 1 : Choisir tous les sommets qui sont à moins de ε de l'arête AB qui doit être fracturée.
Étape 2 : Trier les sommets selon la position de leur projection sur AB : $V_1,...,V_k$.
Étape 3 : Remplacer AB par les arêtes $AV_1$, $V_1V_2,...,V_kB$ et éliminer les doubles arêtes qui en résultent.

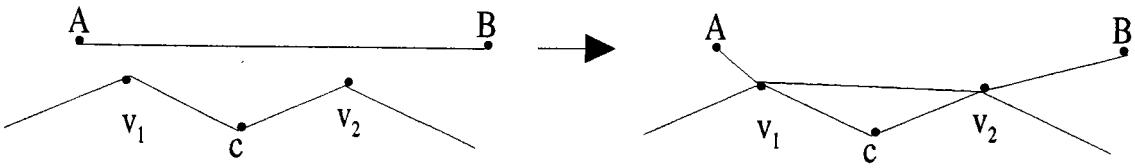



Figure 4 : Éclatement d'arêtes

On remarque facilement que cette méthode perd les propriétés géométriques (un segment est divisé en une suite de segments), de plus l'erreur introduite (qui peut être calculée comme la différence entre l'aire de la région normalisée et l'aire de la région qui ne l'est pas) grandit avec le nombre de normalisations. Les opérations qu'il définit, semblent modifier la valeur du nombre topologique onduleux des points et ce point délicat n'est pas traité, ni même mentionné. En outre, cette méthode est difficile à généraliser à des domaines plus complexes.

•*Méthode de la variable cachée*

V. J. Milenkovic [MIL 88] illustre une méthode appliquée à un problème d'arrangement topologique de n droites représentées par leur équation. La méthode considère les droites comme des courbes ou suite de points *xy-monotones*, c'est à dire monotones par rapport aux axes 0x et 0y (une courbe y(t) (ou une suite de points) est v-monotone si le produit intérieur de v et y(t) est soit non décroissante ou non croissante avec t, plus précisément, la courbe tend à être parallèle à v). Les *variables cachées* du problème sont les formes non représentables des courbes xy-monotones qui approchent les droites. Afin de modéliser les droites, ce système déduit les localisations des sommets et une structure topologique telle que les courbes passent par ces sommets et soient arrangées suivant la topologie. V. J. Milenkovic [MIL 88] présente un exemple de topologie qui permet d'arranger ces droites. Les approximations des droites s'écartent au plus de ε des droites qu'elles représentent, l'erreur introduite dépend du nombre de droites et non pas du nombre d'opérations.
Cette méthode conserve les propriétés géométriques, mais elle est difficile à mettre en oeuvre.

### *4.2.6. Aquarels*

Aquarels est un atelier de qualité numérique pour la réalisation de logiciels scientifiques. Il est issu d'une collaboration entre le CNES, le CEA/DAM, la DGA/DRET, l'IRISA/INRIA et SIMULOG [AQU 96] [BER 95]. Son objectif est de fournir aux concepteurs d'applications

scientifiques des méthodes et des outils pour maîtriser leurs erreurs numériques dans les étapes de développement d'un logiciel FORTRAN 77 :

- lors de la modélisation physique et mathématique, grâce à un outil de calcul formel (intégration du système formel MAPLE),
- lors de l'analyse à priori de la précision des résultats grâce au langage FORTRAN AQUARELS, qui étend FORTRAN 77 par la définition de nouvelles arithmétiques (arithmétique paramétrée : type MULTI PRECISION, arithmétique d'intervalle : type INTERVAL, produit scalaire exact : type DOT PRECISION),
- lors du choix d'une machine cible, grâce à une analyse des qualités numériques de l'unité arithmétique et des fonctions élémentaires (norme IEEE, paramètres primaires ...),
- lors de l'étape de codage, par un contrôle de normes de programmation, par la génération de code, par l'emploi de bibliothèques numériques,
- lors de l'analyse et de la validation, en testant la sensibilité des résultats à une perturbation de l'arithmétique utilisée ou à une perturbation des données [ERH 93].

Cette boîte à outils permet donc de concevoir des logiciels avec une assurance plus grande sur leur fiabilité numérique, grâce à des aides et des tests continus tout au long de leur conception.

## 5. CONCLUSION

La précision des calculs est un problème important, en C.A.O. mais aussi dans les domaines où interviennent les calculs numériques. Les scientifiques essaient d'apporter diverses solutions.

On peut résumer les diverses méthodes sous forme de tableaux :

**Outils mathématiques :**

| SOLUTIONS | AVANTAGES | INCONVENIENTS | REFERENCES |
|---|---|---|---|
| Analyse différentielle | • Connaissance de l'influence des erreurs des données sur les résultats | • Persistance des erreurs<br>• Ne permet d'amélioration | * [STO 93] |
| Permutation-Perturbation | • Obtention d'un intervalle contenant la valeur exacte.<br>• Réduction des effets d'arrondi avec estimation des chiffres significatifs. | • Augmentation du coût en temps et en place mémoire.<br>• Persistance des erreurs.<br>• La comparaison entre deux nombres n'est pas plus fiable. | * [MIC 87]<br>* [MAS 93] |
| Condition-nement d'un problème | • Permet de connaître l'influence des perturbations des données sur les résultats d'un calcul.<br>• Possibilités de trouver de meilleures méthodes de résolutions. | • Ne permet pas d'éviter les problèmes.<br>• Ne résout pas les erreurs introduites par la machine. | * [DAN 89]<br>* [CIA 82]<br>* [HOF 89]<br>* [VIG 74] |
| Différentiation automatique | • Amélioration de la précision dans le calcul de dérivées | • Appliquable qu'aux domaines où les dérivées interviennent | * [EYS96] |

**Techniques de résolutions**

| SOLUTIONS | AVANTAGES | INCONVENIENTS | REFERENCES |
|---|---|---|---|
| Utilisation d'une précision | • Tolérance admise pour déterminer une égalité.<br>• Amélioration partielle des algorithmes.<br>• Possibilité d'améliorer l'estimation en évaluant l'erreur au cours des calculs. | • Difficulté du choix d'un epsilon correct.<br>• Ne résout pas tous les problèmes.<br>• La gestion des erreurs au cours des calculs est onéreuse en temps, espace mémoire ... . | * [MAS 93]<br>* [DAN 89] |
| Arithmétique rationnelle | • Calcul précis.<br>• Suppression des erreurs numériques. | • Représentation perdue au cours des transformations.<br>• Introduction d'erreur pour l'approximation par un rationnel.<br>• Restriction des entiers.<br>• Hausse du temps des calculs. | * [BEN 93]<br>* [HOF 88]<br>* [HOF 89]<br>* [MIC 87]<br>* [MOR 89] |
| Arithmétique paresseuse | • Optimisation du temps de calcul par rapport à l'arithmétique rationnelle traditionnelle.<br>• Précision des calculs pour les cas litigieux. | • Utilisation d'une arithmétique spécialisée.<br>• Augmentation de la place mémoire.<br>• Ses performances dépendent du mode de programmation . | * [BEN 93]<br>* [MOR 89] |
| Arithmétique d'intervalles | • Obtention d'un intervalle contenant la valeur exacte | • Agrandissement rapide des intervalles | * [MUL 89] |
| Représentation symbolique | • Amélioration de la robustesse des algorithmes.<br>•Tests numériques réconfortés par des tests topologiques. | • Ne semble pas s'appliquer directement à des objets curvilignes (courbes...).<br>• Erreurs toujours présentes. | * [HOF 89]<br>* [HOF 88]<br>* [HHK 89] |
| Modifications des données | • Tests numériques exacts | • Pertre des propriétés topologiques | * [MIL 88] |
| Aquarels | • Conception numériquement plus fiable d'algorithmes | • Augmentation du temps | * [AQU 96]<br>* [BER 95] |

On remarque aisément que l'augmentation des précisions demandées entraîne une augmentation du temps et de l'espace mémoire nécessaire. Il faudrait être irréaliste pour penser que l'on peut éviter ces désagréments.

Dans le cadre du projet SACADO du Laboratoire, les méthodes sont actuellement confrontées au problème de la gestion de tolérances, notamment lors de la comparaison de deux réels ou du calcul d'intersection. Il semble alors intéressant de voir comment se propagent les erreurs dans divers algorithmes, afin de trouver une précision meilleure que celle habituellement utilisée pour différents algorithmes.

# CHAPITRE II

# ÉTUDE DES ERREURS COMMISES AU COURS DE L'INTERSECTION DE DEUX DROITES

## 1. INTRODUCTION

Ce chapitre est consacré à l'étude des erreurs commises dans l'algorithme d'intersection de deux droites. Nous avons choisi cet exemple, car cet algorithme intervient dans beaucoup de domaines de la C.A.O. (modélisation, visualisation, lancer de rayon etc...). Cette étude se décompose en deux parties :

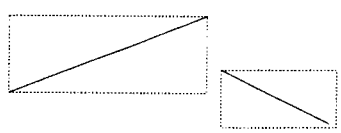
- Une première étude utilise l'analyse différentielle sur les erreurs absolues et les erreurs relatives. On estime aussi l'erreur d'arrondi introduite durant les calculs. Nous obtenons des formules donnant les erreurs sur les résultats en fonction des erreurs sur les données.
- Une deuxième étude utilise l'arithmétique d'intervalle. Nous considérons qu'un réel est représenté par un intervalle auquel il appartient. Cette arithmétique va nous permettre d'obtenir une estimation de la perte de précision en fonction de l'angle.

Nous commençons par rappeler les algorithmes intervenant au cours de l'intersection de deux segments.

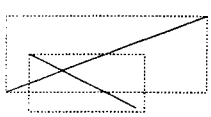
## 2. PRÉSENTATION DE L'ALGORITHME D'INTERSECTION

Soient deux segments S1 et S2 définis par leurs sommets : $P1(x_1,y_1)$, $P2(x_2,y_2)$, $P3(x_3,y_3)$ et $P4(x_4,y_4)$.

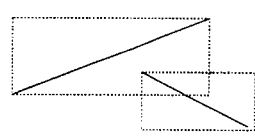
Une manière simple de calculer le point d'intersection P(x,y) des deux segments, est de déterminer l'intersection des droites supports des segments et de vérifier si le point calculé appartient bien aux deux segments. Mais ce procédé effectue ses tests d'appartenance sur des valeurs déterminées par calculs et donc sujettes à des erreurs.



cas 1: pas d'intersection



cas 2: possibilité d'intersection :
il est nécessaire de faire d'autres calculs



cas 3: situation identique au cas 2

Figure 5

Remarque : On peut éviter certains calculs inutiles en effectuant des tests antérieurs à ce traitement, qui élimineront de façon sûre une partie des cas où les segments ne risquent pas de se couper. Le test sur les rectangles englobants est un bon filtre [GAR 88], il élimine un nombre important de cas, même s'il ne les supprime pas tous (cf figure 5).

Dans le cas d'une intersection éventuelle, on va calculer les coefficients des deux droites portant les segments : $a_i$, $b_i$, $c_i$ pour le segment $S_i$ (i=1, 2).

$a_1 = y_1 - y_2$; $b_1 = x_2 - x_1$; $c_1 = -a_1 \times x_1 - b_1 \times y_1$.

$a_2 = y_3 - y_4$; $b_2 = x_4 - x_3$; $c_2 = -a_2 \times x_3 - b_2 \times y_3$.

Ensuite, on recherche le point d'intersection P des deux droites et on regarde si le point appartient aux deux segments. On a donc le système suivant : $\begin{cases} a_1x + b_1y + c_1 = 0 \\ a_2x + b_2y + c_2 = 0 \end{cases}$ à résoudre.

Pour vérifier l'appartenance aux deux segments du point d'intersection, on a à notre disposition deux possibilités :

- Regarder si le point P appartient à chacun des segments, pour cela on compare les distances PP1 + PP2 avec P1P2 et PP3 + PP4 avec P3P4, si on a égalité dans les deux cas, alors le point est bien le point d'intersection des deux segments, sinon, les segments ne se coupent pas. Cette méthode fait intervenir des calculs supplémentaires, donc une augmentation du temps de calcul et une amplification des erreurs, ce qui peut occasionner des erreurs d'exécution de l'algorithme.
- On peut tout simplement vérifier que les coordonnées du point d'intersection se situe entre les coordonnées des quatre points. On n'a alors aucun calcul supplémentaire, la réponse n'est basée que sur des tests, d'où un risque moins élevé d'erreurs. Ce qui correspond au test suivant:

si $(x_1 \leq x \leq x_2)$ et $(x_3 \leq x \leq x_4)$ et $(y_1 \leq y \leq y_2)$ et $(y_3 \leq y \leq y_4)$

alors les segments se coupent.

Le deuxième moyen est celui qu'il faut utiliser, car on sait déjà que le point appartient aux droites, mais le premier est intéressant dans la mesure où il permet d'étudier également une fonction de base (appartenance d'un point à un segment) et comme le modèle Sacado utilise cette solution, nous employons ce procédé pour que nos tests comparatifs soient significatifs. Il est évident qu'il faut modifier ce test dans le modèle Sacado.

Les algorithmes correspondants à ces divers points sont détaillés en annexe 2.

## 3. ANALYSE DIFFERENTIELLE

### 3.1. Erreurs absolues

Pour effectuer cette étude, nous utilisons l'analyse différentielle que l'on a présentée au chapitre I § 4.1.1, appliquée aux erreurs absolues. Dans notre cas, les $f_i$ correspondent aux calculs des x, y, $d_1$, $d_2$, $d_3$ et les variables $x_i$ aux coordonnées $x_i$ et $y_i$ (i=1,..., 4) du problème.

Soient $e_{xi}$ l'erreur sur le nombre $x_i$ et $e_{yi}$ l'erreur sur le nombre $y_i$ ( i = 1,..,4). Nous obtenons :

$$e_x = [\, e_{x1}\{(-x_3y_4+y_3x_4-y_2b_2)\times det - n1\, a_2\} + e_{x2}\{(y_4x_3-x_4y_3+y_1b_2)\times det + n1\, a_2\} + e_{x3}\{(y_4b_1+y_2x_1-x_2y_1)\times det + n1\, a_1\} + e_{x4}\{(-y_3b_1-x_1y_2+x_2y_1)\times det - n1\, a_1\} + e_{y1}\{x_2\, b_2\times det - n1\, b_2\} + e_{y2}\{-x_1\, b_2\times det + n1\, b_2\} + e_{y3}\{-x_4\, b_1\times det + n1\, b_1\} + e_{y4}\{x_3\, b_1\times det - n1\, b_1\}\,] / det^2$$

de même :

$$e_y = [\, e_{x1}\{y_2\, a_2\times det - n2\, a_2\} + e_{x2}\{-y_1\, a_2\times det + n2\, a_2\} + e_{x3}\{-y_4\, a_1\times det + n2\, a_1\} + e_{x4}\{y_3\, a_1\times det - n2\, a_1\} + e_{y1}\{(-x_2a_2-x_3y_4+x_4y_3)\times det - n2\, b_2\} + e_{y2}\{(x_1a_2+x_3y_4-x_4y_3)\times det + n2\, b_2\} + e_{y3}\{(x_1y_2-x_2y_1+x_4a_1)\times det + n2\, b_1) + e_{y4}\{(-x_1y_2+x_2y_1-x_3a_1)\times det - n2\, b_1\}\,] / det^2$$

où $n1 = c_2 \times b_1 - c_1 \times b_2$, $n2 = c_1 \times a_2 - a_1 \times c_2$, avec $x = \frac{n1}{det}$ et $y = \frac{n2}{det}$.

Pour le déterminant, Nous avons :

$$e_{det} = a_2(e_{x1} - e_{x2}) + a_1(e_{x4} - e_{x3}) + b_2(e_{y1} - e_{y2}) + b_1(e_{y4} - e_{y3})$$

En ce qui concerne les distances, nous trouvons :

$$e_{d1} = \frac{(x-x_1)\times(e_x-e_{x1}) - (y-y_1)\times(e_y-e_{y1})}{d_1}$$

$$e_{d2} = \frac{(x-x_2)\times(e_x-e_{x2}) - (y-y_2)\times(e_y-e_{y2})}{d_2}$$

$$e_{d3} = \frac{(x_2-x_1)\times(e_{x2}-e_{x1}) - (y_2-y_1)\times(e_{y2}-e_{y1})}{d_3}$$

### 3.2. Erreurs relatives

Nous reprenons ici aussi l'analyse différentielle, appliquée aux erreurs relatives.
Soient $er_{xi}$ l'erreur sur le nombre $x_i$ et $er_{yi}$ l'erreur sur le nombre $y_i$ ( i = 1,..,4). On obtient :

$$
\begin{aligned}
er_x = {} & er_{x1}\left(\frac{-x_1x_3y_4 + x_1x_4y_3 - x_1y_2b_2}{n1} + \frac{-x_1a_2}{det}\right) \\
& + er_{x2}\left(\frac{x_2x_3y_4 - x_2x_4y_3 + x_2y_1b_2}{n1} + \frac{x_2a_2}{det}\right) \\
& + er_{x3}\left(\frac{x_3y_4b_1 + x_1x_3y_2 - x_2x_3y_1}{n1} + \frac{x_3a_1}{det}\right) \\
& + er_{x4}\left(\frac{-x_4y_3b_1 - x_1x_4y_2 + x_2x_4y_1}{n1} + \frac{-x_4a_1}{det}\right) \\
& + er_{y1}\left(\frac{x_2y_1b_2}{n1} + \frac{-y_1b_2}{det}\right) + er_{y2}\left(\frac{-x_1y_2b_2}{n1} + \frac{y_2b_2}{det}\right) \\
& + er_{y3}\left(\frac{-x_4y_3b_1}{n1} + \frac{y_3b_1}{det}\right) + er_{y4}\left(\frac{x_3y_4b_1}{n1} + \frac{-y_4b_1}{det}\right)
\end{aligned}
$$

$$
\begin{aligned}
er_y = {} & er_{x1}\left(\frac{x_1y_2a_2}{n2} - \frac{x_1a_2}{det}\right) + er_{x2}\left(\frac{-x_2y_1a_2}{n2} + \frac{x_2a_2}{det}\right) \\
& + er_{x3}\left(\frac{-x_3y_4a_1}{n2} + \frac{x_3a_1}{det}\right) + er_{x4}\left(\frac{x_4y_3a_1}{n2} - \frac{x_4a_1}{det}\right) \\
& + er_{y1}\left(\frac{-x_2y_1a_2 + x_4y_1y_3 - x_3y_1y_4}{n2} - \frac{y_1b_2}{det}\right) \\
& + er_{y2}\left(\frac{x_1y_2a_2 + x_3y_2y_4 - x_4y_2y_3}{n2} + \frac{y_2b_2}{det}\right) \\
& + er_{y3}\left(\frac{x_1y_2y_3 - x_2y_1y_3 + x_4y_3a_1}{n2} + \frac{y_3b_1}{det}\right) \\
& + er_{y4}\left(\frac{x_2y_1y_4 - x_1y_2y_4 - x_3y_4a_1}{n2} - \frac{y_4b_1}{det}\right)
\end{aligned}
$$

$$
\begin{aligned}
er_{det} = {} & \frac{1}{det}\left\{a_2\left(x_1er_{x1} - x_2er_{x2}\right) + a_1\left(x_4er_{x4} - x_3er_{x3}\right)\right. \\
& \left. + b_2\left(y_1er_{y1} - y_2er_{y2}\right) + b_1\left(y_4er_{y4} - y_3er_{y3}\right)\right\}
\end{aligned}
$$

$$
er_{d1} = \frac{1}{d_1^2}\left\{\left(er_x.x - er_{x1}.x_1\right)\left(x - x_1\right) + \left(er_y.y - er_{y1}.y_1\right)\left(y - y_1\right)\right\}
$$

$$
er_{d2} = \frac{1}{d_2^2}\left\{\left(er_x.x - er_{x2}.x_2\right)\left(x - x_2\right) + \left(er_y.y - er_{y2}.y_2\right)\left(y - y_2\right)\right\}
$$

$$
er_{d3} = \frac{1}{{d_3}^2}\left\{-er_{x1}x_1b_1 + er_{y1}y_1a_1 + er_{x2}x_2b_1 - er_{y2}y_2a_1\right\}
$$

### 3.3. Erreurs d'arrondi

Si x est un nombre réel, il sera représenté sous une forme approchée dans la mémoire de l'ordinateur. La notation la plus utilisée actuellement est celle de la représentation en virgule flottante, définie au paragraphe 3 du chapitre I.
Cette représentation entraîne que la précision dans l'approximation d'un nombre réel est toujours une précision relative [HOF 89] :

$$\frac{ea_x}{x} = \frac{ea_m}{m} \leq \frac{b^{-N}}{b^{-1}} = b^{1-N},$$

où $ea_{terme}$ est l'erreur d'arrondi sur "terme".
On note $\varepsilon = b^{1-N}$ cette précision relative.

D'après J. P. Demailly [DEM 91], l'erreur d'arrondi sur une somme est proportionnelle à la précision relative, et on a:

$$ea_{(x+y)} \leq \varepsilon\,(|x| + |y|),$$

de même, pour l'erreur sur un produit :

$$ea_{(x \times y)} \leq \varepsilon\,(|x \times y|).$$

Pour notre algorithme, nous tenons compte des erreurs d'arrondi sur les résultats finaux et les résultats intervenant dans des comparaisons. On obtient alors :

$$\boxed{ea_{det} = \varepsilon\left(|a_1 b_2| + |a_2 b_1|\right)}$$

$$\boxed{ea_x = \varepsilon\left(\frac{|c_2 b_1| + |c_1 b_2|}{|a_1 b_2| + |a_2 b_1|}\right)}$$

$$\boxed{ea_y = \varepsilon\left(\frac{|c_1 a_2| + |c_2 a_1|}{|a_1 b_2| + |a_2 b_1|}\right)}$$

$\boxed{ea_{di} = \varepsilon d_i}$ pour i = 1,...,3

### 3.4. Étude expérimentale

Nous reprenons l'exemple du pentagone décrit au paragraphe 2 du chapitre I. Cet exemple, basé exclusivement sur l'intersection de droites, met en évidence les problèmes de l'arithmétique flottante. On a calculé, pour chaque cas, l'erreur "exacte", correspondant à l'erreur absolue déterminée à l'aide de l'analyse différentielle (Tableau 3, 4). Le test sur le déterminant a été effectué par rapport à l'erreur "exacte" introduite plus l'erreur d'arrondi. L'erreur de départ a été initialisée à l'erreur absolue de la machine c'est à dire $10^{-11}$. Les résultats des tableaux sont les erreurs maximales sur les coordonnées des cinq points.

Tableau 3 : Erreurs "exactes" pour les opérations "going in", "going out".

| p | $out^1(in^1())$ | $out^2(in^2())$ | $out^3(in^3())$ | $out^4(in^4())$ |
|---|---|---|---|---|
| 0,1 | 2,10E-10 | 2,69E-09 | 4,08E-09 | 1,06E-08 |
| 0,01 | 2,12E-10 | 2,06E-08 | 2,00E-08 | 1,02E-07 |
| 0,001 | 2,10E-10 | 2,00E-07 | 1,94E-07 | 1,83E-04 |
| 0,0001 | 2,10E-10 | 2,00E-06 | 2,21E-05 | 7,02E-01 |
| 0,00001 | 2,10E-10 | 1,90E-05 | 6,71E-02 | (det=0) |

Tableau 4 : Erreurs "exactes" pour les opérations "going out", "going in".

| p | $in^1(out^1())$ | $in^2(out^2())$ | $in^3(out^3())$ | $in^4(out^4())$ |
|---|---|---|---|---|
| 0,1 | 3,95E-10 | 1,04E-09 | 2,79E-09 | 3,59E-09 |
| 0,01 | 4,08E-10 | 1,10E-09 | 1,66E-08 | 1,53E-08 |
| 0,001 | 4,09E-10 | 1,10E-09 | 1,01E-07 | 2,35E-06 |
| 0,0001 | 4,10E-10 | 1,10E-09 | 2,19E-06 | 5,44E-02 |
| 0,00001 | 4,11E-10 | 1,10E-09 | (det=0) | (det=0) |

Pour n = 1, l'erreur calculée reste toujours de même ordre ( $\approx 10^{-10}$ ), tandis que pour n > 1, l'erreur augmente avec n et la diminution de p. Ceci semble normal, puisqu'il est effectué plus de calculs et le fait que p deviennent petit, entraîne plus de perte de bits significatifs.
Nous voyons aussi que les erreurs augmentent plus quand l'opération "going in" est effectuée en premier, car les points deviennent proches entre eux et les nombres deviennent proche de zéro.
Par rapport aux erreurs réellement commises, celles calculées ici, sont, pour la plupart, inférieures, surtout pour de petites valeurs de p.

De la même manière, on a calculé l'erreur relative pour chaque cas (Tableau 5, 6). Le test sur le déterminant a été effectué par rapport à l'erreur relative fois le déterminant plus l'erreur d'arrondi. L'erreur de départ a été initialisée à l'erreur relative de la machine, soit $10^{-10}$. Les résultats des tableaux sont les erreurs relatives maximales multipliées par les résultats finaux correspondants, afin d'obtenir des erreurs absolues pour pouvoir comparer avec les valeurs des erreurs réelles.

Tableau 5 : Erreurs relatives pour les opérations "going in", "going out".

| p | $out^1(in^1( ))$ | $out^2(in^2( ))$ | $out^3(in^3( ))$ | $out^4(in^4( ))$ |
|---|---|---|---|---|
| 0,1 | 3,30E-10 | 2,78E-09 | 4,09E-09 | 1,06E-08 |
| 0,01 | 3,03E-10 | 2,07E-08 | 2,01E-08 | 5,85E-07 |
| 0,001 | 3,00E-10 | 2,00E-07 | 2,64E-05 | 3,34E-03 |
| 0,0001 | 3,00E-10 | 2,00E-06 | (det=0) | (det=0) |
| 0,00001 | 3,00E-10 | 1,97E-05 | (det=0) | (det=0) |

Tableau 6 : Erreurs relatives pour les opérations "going out", "going in".

| p | $in^1(out^1( ))$ | $in^2(out^2( ))$ | $in^3(out^3( ))$ | $in^4(out^4( ))$ |
|---|---|---|---|---|
| 0,1 | 4,85E-10 | 1,13E-09 | 2,78E-09 | 3,58E-09 |
| 0,01 | 4,98E-10 | 1,19E-09 | 1,16E-08 | 1,54E-08 |
| 0,001 | 4,99E-10 | 1,19E-09 | 1,01E-07 | 4,47E-06 |
| 0,0001 | 4,99E-10 | 1,19E-09 | 1,67E-06 | 8,83E-02 |
| 0,00001 | 4,99E-10 | 1,19E-09 | (det=0) | (det=0) |

Les erreurs relatives ont le même comportement que les erreurs "exactes". Les différences entre les erreurs relatives et les erreurs réelles varient de $\pm 10^{-2}$ au plus. Elles deviennent inférieures dès que les valeurs de p deviennent petites et pour $n > 2$.
Les erreurs relatives semblent majorer plus rapidement le déterminant que les erreurs "exactes" pour l'opération "going in".

Nous remarquons que les erreurs relatives et les erreurs "exactes" sont à peu près du même ordre de grandeur. La première devient plus grande que pour $n > 2$ et $p < 10^{-4}$ pour l'opération $out^n(in^n( ))$. Ces différences entre les erreurs réelles et les erreurs calculées peuvent venir de fait que l'erreur initiale est imposée et n'est pas forcément (sûrement) la bonne. Il est très difficile d'obtenir l'erreur absolue réelle d'un nombre sur un ordinateur. De même, l'erreur relative de chaque réel n'est pas exactement celle due à l'arrondi de l'ordinateur. De plus, cette analyse ne tient pas compte des erreurs commises par l'ordinateur au cours des exécutions, elle ne permet que d'étudier la propagation d'une erreur que l'on a au départ, d'où une différence importante entre les résultats. C'est pour éviter ces désagréments, qu'une étude à l'aide d'intervalles a été mise en oeuvre au paragraphe 4 de ce chapitre.

**Remarque :**

Dans notre exemple, on a initialisé l'erreur relative de départ à l'erreur relative de la machine. En modifiant l'écriture du calcul des erreurs relatives en remplaçant les erreurs $er_{xi}$ par l'erreur machine, on obtient :

$$
\begin{aligned}
er_x = {} & \frac{1}{n_1}\cdot\Big((x_3y_4 - x_3y_3 + x_3y_3 - x_4y_3)\cdot(x_2er_{x2} - x_1er_{x1}) + \\
& (x_2y_1 - x_1y_1 + x_1y_1 - x_1y_2)\cdot(x_4er_{x4} - x_3er_{x3})\Big) + \\
& \frac{1}{n_1}\cdot\Big[b_1\Big((er_{x3} + er_{y4})\cdot(x_3y_4 - x_3y_3) + (er_{x4} + er_{y3})\cdot(x_3y_3 - x_4y_3)\Big) + \\
& b_2\Big((er_{x2} + er_{y1})\cdot(x_2y_1 - x_1y_1) + (er_{x1} + er_{y2})\cdot(x_1y_1 - x_1y_2)\Big)\Big] + \\
& \frac{1}{det}\Big(a_2\cdot(x_2er_{x2} - x_1er_{x1}) + a_1\cdot(x_3er_{x3} - x_4er_{x4}) + \\
& b_2\cdot(y_2er_{y2} - y_1er_{y1}) + b_1\cdot(y_3er_{y3} - y_4er_{y4})\Big) + \\
& \frac{1}{n_1}\cdot\Big(x_3y_3(er_{x3} + er_{y4} - er_{x4} - er_{y3}) + x_1y_1(er_{x2} + er_{y1} - er_{x1} - er_{y2})\Big)
\end{aligned}
$$

Si $er_{xi} = er_{yi} = \varepsilon$, alors :

$$er_x = \varepsilon\cdot\frac{n_1}{n_1} + 2\varepsilon\cdot\frac{n_1}{n_1} - 2\varepsilon\frac{det}{det} = \varepsilon$$

On remarque alors, que l'angle formé par deux droites n'intervient pas ici; ce qui en pratique, est faux, car on sait que ce calcul d'intersection de droites est mal conditionné pour des angles petits. Ceci met en évidence le fait que le calcul sur ordinateur introduit des erreurs supplémentaires.

### 3.5. Conclusion

On constate dans l'exemple, que ce soit pour l'erreur "exacte" ou pour l'erreur relative, que l'estimation est en générale trop petite, mais qu'elle a le même comportement que l'erreur réelle. Une majoration des formules a été effectuée, mais les résultats de celle-ci étaient supérieurs aux erreurs réelles et avaient tendance à diverger, c'est pourquoi cette étude n'est ici que citée.

Nous comprenons mieux pourquoi les résultats sont inférieurs à la réalité à l'aide de la remarque. Celle-ci nous montre bien qu'une analyse différentielle n'est pas suffisante pour obtenir une bonne estimation des erreurs affectant les résultats finaux obtenus à l'aide d'un ordinateur.

Nous savons que les erreurs numériques augmentent rapidement quand l'angle entre les deux segments est faible. Nous utilisons alors l'arithmétique d'intervalle pour déterminer l'erreur maximale commise lors de l'intersection. Nous obtenons aussi une estimation de l'erreur affectant le résultat en fonction de l'angle.

## 4. ARITHMETIQUE D'INTERVALLES

### 4.1. Étude théorique

Nous allons utiliser l'arithmétique d'intervalles décrite au paragraphe 4.2.3 du chapitre I, pour l'intersection de segments. Il faut donc encadrer chaque valeur par deux bornes représentables en machines. De plus, on sait que l'erreur sur un réel x quelconque est borné par $\varepsilon.|x|$. Chaque réel sera donc représenté par l'intervalle $[x - \varepsilon |x|, x + \varepsilon |x|]$.

L'erreur relative sera notée $\varepsilon$ afin de laisser les expressions simples et parlantes. Elle est définie en fonction de la précision de l'ordinateur et est indépendante du nombre auquel elle est liée ( $\varepsilon > 0$ ).

Dans tous les calculs effectués, on supposera que $\varepsilon^2.(...) \approx 0$, c'est pourquoi ce terme a été omis ainsi que les degrés supérieurs.

Les intervalles des différents réels considérés sont les suivants :

$x_i - |x_i|\varepsilon \leq x_i \leq x_i + |x_i|\varepsilon$

$y_i - |y_i|\varepsilon \leq y_i \leq y_i + |y_i|\varepsilon$

**Calcul du déterminant :**

On obtient pour le déterminant ( Annexe 5) :

$$\boxed{\det - \varepsilon(M_9 + M_{10}) \leq \det \leq \det + \varepsilon(M_9 + M_{10})}$$

Soit une erreur absolue $e_{det} = \varepsilon(M_9+M_{10})$ ou une erreur relative $er_{det} = \varepsilon(M_9+M_{10})/det$.

Le déterminant sera considéré comme nul si zéro appartient à l'intervalle, soit :

$det - \varepsilon(M_9+M_{10}) \leq 0 \leq det + \varepsilon(M_9+M_{10})$

**Calcul de l'abscisse x :**

On obtient après calcul ( Annexe 3 ), un encadrement de l'abscisse x :

$$\boxed{\mathbf{x - \varepsilon.M_{11} \leq x \leq x + \varepsilon.M_{11}}}$$

Soit une erreur absolue $e_x = \varepsilon.M_{11}$, ou une erreur relative $er_x = \varepsilon.M_{11}/x$ ( $x \neq 0$ ).

**Calcul de l'ordonnée y :**

On obtient après calcul ( Annexe 4 ), un encadrement de l'ordonnée y :

$$\boxed{\mathbf{y - \varepsilon.M_{12} \leq y \leq y + \varepsilon.M_{12}}}$$

Soit une erreur absolue $e_y = \varepsilon.M_{12}$, ou une erreur relative $er_y = \varepsilon.M_{12}/y$ ( $y \neq 0$ ).

Nous avons donc obtenu un encadrement théorique du point d'intersection en fonction de l'erreur relative de l'ordinateur.

**Calcul de la distance $d_1$ :**

$$d_1 = \sqrt{(x-x_1)^2 - (y-y_1)^2}$$

Les calculs sont détaillés dans l'annexe 6.

En posant :

$$A = (x - x_1)^2 + (y-y_1)^2 - \varepsilon (M_{13} + M_{14})$$
$$B = (x - x_1)^2 + (y-y_1)^2 + \varepsilon (M_{13} + M_{14})$$

On obtient donc dans le cas $d_1 \neq 0$

$$\sqrt{(x-x_1)^2 + (y-y_1)^2 - \varepsilon(M_{13} + M_{14})} \leq d_1 \leq \sqrt{(x-x_1)^2 + (y-y_1)^2 + \varepsilon(M_{13} + M_{14})}$$

**Calcul de la distance $d_2$ :**

Avec le même raisonnement, on calcule l'encadrement pour $d_2$ :

C et D sont les valeurs sous les racines des bornes de l'intervalles de $d_2$ :

$$C = (x - x_2)^2 + (y-y_2)^2 - \varepsilon (M_{15} + M_{16})$$
$$D = (x - x_2)^2 + (y-y_2)^2 + \varepsilon (M_{15} + M_{16})$$

où

$$M_{15} = 2|x - x_2|(M_{11} + |x_2|)$$
$$M_{16} = 2|y_2 - y_1|(M_{12} + |y_2|)$$

Si $C \leq 0 \leq D$

alors $d_2 = 0$

$$C = D = 0$$

sinon $\sqrt{C} \leq d_2 \leq \sqrt{D}$

On obtient donc dans le cas $d_2 \neq 0$

$$\sqrt{(x-x_2)^2 + (y-y_2)^2 - \varepsilon(M_{15} + M_{16})} \leq d_2 \leq \sqrt{(x-x_2)^2 + (y-y_2)^2 + \varepsilon(M_{15} + M_{16})}$$

**Calcul de la distance $d_3$ :**

Même principe, E, F étant les valeurs sous les racines des bornes de l'intervalles de $d_3$ :

$$E = (x_2 - x_1)^2 + (y_2-y_1)^2 - \varepsilon (M_{17} + M_{18})$$
$$F = (x_2 - x_1)^2 + (y_2-y_1)^2 + \varepsilon (M_{17} + M_{18})$$

où

$$M_{17} = 2(|x_1| + |x_2|)|x_2 - x_1|$$

$$M_{18} = 2(|y_1| + |y_2|)|y_2 - y_1|$$

Si $E \leq 0 \leq F$

alors $d_3 = 0$

$E = F = 0$

sinon $\sqrt{E} \leq d_3 \leq \sqrt{F}$

On obtient donc dans le cas $d_3 \neq 0$

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 - \varepsilon(M_{17} + M_{18})} \leq d_3 \leq \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \varepsilon(M_{17} + M_{18})}$$

On obtient donc pour les distances $d_1$, $d_2$, $d_3$ :

**si $A \leq 0 \leq B$ alors $d_1 = 0$, $A = 0$, $B = 0$**
**sinon $\sqrt{A} \leq d_1 \leq \sqrt{B}$**
**si $C \leq 0 \leq D$ alors $d_2 = 0$, $C = 0$, $D = 0$**
**sinon $\sqrt{C} \leq d_2 \leq \sqrt{D}$**
**si $E \leq 0 \leq F$ alors $d_3 = 0$, $E = 0$, $F = 0$**
**sinon $\sqrt{E} \leq d_3 \leq \sqrt{F}$**

et on aura :

**$d_1 + d_2 = d_3$ si et seulement si $\sqrt{A} + \sqrt{C} - \sqrt{F} \leq 0 \leq \sqrt{B} + \sqrt{D} - \sqrt{E}$**

### 4.2. Études expérimentales

**Introduction :**

Nous avons effectué deux expériences à partir de l'algorithme de l'intersection de droites. Avec la première, nous étudions le comportement de l'arithmétique d'intervalle pour des intersections de droites et nous comparons les résultats avec ceux obtenus par un logiciel déjà existant tel que Sacado.

La deuxième nous sert à déterminer l'influence de l'angle que forment les deux droites sur les résultats.

**Étude 1 :**

Nous utilisons le segment d'extrémités [-20,-20] et [30,30], et nous effectuons des rotations ayant pour centre le milieu du segment et des angles de plus en plus petits, car le risque d'erreurs dans le calcul d'intersection de segments (cas particulier des droites) est plus

important quand l'angle entre les deux segments est proche de 0° ou 180°, cela étant dû au mauvais conditionnement de la matrice d'inversion. On obtient alors différents segments très proches du segment initial (parfois invisibles sur l'écran) et on calcule les intersections, entre le segment initial et ceux construits par rotations, avec le modèle Sacado et le modèle sur le calcul d'intervalles.

Les intervalles que l'on utilise ici sont ceux déterminés à partir de l'erreur relative de la machine qui est $\varepsilon$ = 1E-14, les algorithmes correspondants sont détaillés en annexe 7.

Le résultat théorique de chacune des intersections est évidement le centre de rotation de coordonnées [5,5], et ceci quel que soit l'angle.

**Résultats pour Sacado :**

| ANGLE | INTERSECTION | EX | EY |
|---|---|---|---|
| 1E-02° | oui | +1.11E-12 | +1.11E-12 |
| 1E-03° | superposées | | |
| 1E-04° | superposées | | |
| 1E-05° | superposées | | |
| 1E-06° | superposées | | |
| 1E-07° | superposées | | |

Les erreurs EX et EY représentent la différence entre le résultat de l'intersection et le résultat théorique. Nous voyons facilement que Sacado considère deux segments superposés (det < $\varepsilon$) dès que l'angle entre ces deux derniers est plus petit que 1E-02. Ceci est du à la tolérance choisie pour décider si le déterminant est nul ou non, car elle est trop importante pour ce cas.

**Résultats pour l'algorithme basé sur les intervalles :**

| ANGLE | INTERSECTION | EX1 | EY1 | EX2 | EY2 |
|---|---|---|---|---|---|
| 1E-02° | oui | -1,3E-12 | -1,3E-12 | 5,15E-09 | 5,15E-09 |
| 1E-03° | oui | -1,303E-11 | -1,303E-11 | 5,15E-08 | 5,15E-08 |
| 1E-04° | oui | -3,91E-10 | -3,91E-10 | 5,15E-07 | 5,15E-07 |
| 1E-05° | oui | -6.212E-09 | -6,212E-09 | 5,15E-06 | 5,15E-06 |
| 1E-06° | oui | -2.61E-08 | -2,61E-08 | 5,15E-05 | 5,15E-05 |
| 1E-07° | oui | +5.211E-07 | +5,211E-07 | 5,15E-04 | 5,15E-04 |

Les erreurs EX1, EY1 représentent la différence entre le résultat de l'intersection et le résultat théorique, et EX2, EY2, les erreurs absolues calculées à l'aide des intervalles.

Par rapport à Sacado, cet algorithme détecte l'intersection pour de petits angles. On remarque aussi que dans ces cas, la précision du point calculé diminue avec l'angle. L'égalité entre les erreurs sur les abscisses et sur les ordonnées est seulement dû au fait que les valeurs recherchées sont identiques.

Si on se ramène aux erreurs relatives (erreur absolue / valeur calculée), on voit que celle-ci devient très grande (≈1E-07) pour de petits angles de rotation et que le rapport entre l'erreur réelle et celle calculée est d'environ 1E-03. L'estimation des erreurs est donc supérieure à la réalité, mais elle ne semble pas affaiblir le bon fonctionnement de l'algorithme, puisque, bien que celui-ci ait ses tests basés sur ces estimations, il réussit à considérer l'intersection et non pas la superposition des segments comme Sacado.

**Étude 2 :**

*•Introduction*

Nous effectuons une étude statistique portant sur une série de 10 segments aléatoires. A chacun de ces segments, nous faisons subir une suite de rotations de centre, le milieu du segment, et d'angles variant entre 90° et $10^{-5}$°.

Pour chaque angle, nous obtenons un échantillon d'une vingtaine d'estimations d'erreurs (abscisses et oordonnées). Nous ne prenons pas en compte les cas où une valeur calculée, durant l'intersection, a été forcée à 0 (0 ∈ intervalle calculé), car cela fausserait les résultats, puisque l'erreur a, elle aussi, été mise à 0.

A partir de ces échantillons, nous procédons à une étude statistique simple : moyenne, variance et covariance.

Dans un premier temps, nous calculons les intervalles exacts, c'est à dire sans simplification des "max" et des "min" intervenants dans les opérations (voir annexes 3 à 6), ensuite on utilise les formules simplifiées.

*•Résultats sans majoration*

Les variables de chaque échantillon ne sont pas énumérées, seuls la moyenne et l'écart type pour chaque angle sont ici cités.

Notations :

m : moyenne

σ : écart type

- **90°** : m = 1,133E-13 et σ = 1,195E-15
- **60°** : m = 1,365E-13 et σ = 1,755E-14
- **45°** : m = 2,020E-13 et σ = 9,978E-15

- **30°** : m = 1,899E-13 et $\sigma$ = 2,245E-15
- **10°** : m = 2,000E-12 et $\sigma$ = 3,118E-13
- **5°** : m = 2,476E-12 et $\sigma$ = 5,813E-13
- **1°** : m = 7,544E-12 et $\sigma$ = 3,118E-12
- **0,1°** : m = 5,993E-11 et $\sigma$ = 9,844E-12
- **0,01°** : m = 2,240E-09 et $\sigma$ = 3,070E-10
- **0,001°** : m = 6,911E-09 et $\sigma$ = 9,337E-10
- **0,0001°** : m = 5,914E-08 et $\sigma$ = 3,025E-09
- **0,00001°** : m = 6,900E-07 et $\sigma$ = 3,920E-09

Nous voyons très bien que pour les angles compris entre 90° et 10° (exclu), l'erreur relative calculée est toujours du même ordre de grandeur (≈1E-13). L'erreur relative de la machine étant de 1E-14, nous constatons que la perte de précision est quasi nulle, puisque les intervalles majorent les erreurs.

Entre 10° et 0,1°, l'estimation augmente de 1/10, la précision des calculs est moins bonne, mais reste encore correcte.

Ensuite, à partir de 0,1°, l'erreur relative commise ne cesse d'augmenter et pour 0,00001°, on va jusqu'à perdre la moitié de la précision de départ.

Ces résultats sont cohérents avec la réalité, puisque, quand l'angle entre deux segments devient petit, la matrice du système à résoudre est très mal conditionnée. Nous obtenons des majorations des erreurs relatives réelles, mais ces majorations ne semblent pas excessives, car l'algorithme détecte toujours l'intersection.

Nous remarquons que la dispersion entre les moyennes et les séries des échantillons n'est pas considérable. Nous déterminons donc l'erreur relative sur le calcul d'intersection suivant l'angle entre les deux segments :

| | | |
|---|---|---|
| **90°-11°** | **: erreur ≈ 1E-13** | **perte de : 1 chiffre significatif** |
| **10°-1°** | **: erreur ≈ 1E-12** | **perte de : 2 chiffres significatifs** |
| **0,1°** | **: erreur ≈ 1E-11** | **perte de : 3 chiffres significatifs** |
| **0,01** | **: erreur ≈ 1E-09** | **perte de : 5 chiffres significatifs** |
| **0,001** | **: erreur ≈ 1E-09** | **perte de : 5 chiffres significatifs** |
| **0,0001** | **: erreur ≈ 1E-08** | **perte de : 6 chiffres significatifs** |
| **0,00001** | **: erreur ≈ 1E-07** | **perte de : 7 chiffres significatifs** |

•*Résultats avec majoration*

- **5°** : m = 7,368E-10 et $\sigma$ = 7,524E-10
- **1°** : m = 1,939E-09 et $\sigma$ = 2,208E-09
- **0,1°** : m = 2,216E-08 et $\sigma$ = 2,143E-08
- **0,01°** : m = 2,169E-07 et $\sigma$ = 2,174E-07
- **0,001°** : m = 2,513E-06 et $\sigma$ = 3,490E-06
- **0,0001°** : m = 2,907E-05 et $\sigma$ = 2,975E-05
- **0,00001°** : m = 1,410E-04 et $\sigma$ = 1,256E-04

Entre 90° et 10°, les calculs n'ont pas été effectués, car pour ces angles, l'erreur n'est pas importante.

Les majorations intervenants dans la simplification des "max" et des "min" entraînent une augmentation de l'ordre de 1E+03. Mais le comportement des estimations est identique au précédent. Cette majoration permet un temps de calcul moins élevé, puisqu'il est effectué moins de comparaisons, toutefois pour pouvoir utiliser cette majoration, il faudra tenir compte de cette augmentation de l'estimation.

### 4.3. Conclusion

L'étude statistique effectuée, permet de déterminer une estimation de la perte de précision en fonction de l'angle. Ce résultat paraît intéressant, car dans les modèles de C.A.O, les epsilons utilisés sont souvent donnés par expérience, mais rien ne permet de savoir s'ils sont corrects dans tous les cas de figures.

Cette étude a été effectuée à partir de valeurs ayant comme erreur relative de départ, celle de la machine. Mais on sait que si on utilise des points déterminés par des calculs intermédiaires, l'erreur relative correspondante à chacun de ces points est différente.

Les premiers résultats montrent bien que l'erreur calculée est supérieure à l'erreur réelle ( $\approx$ 1E+03 ), ce qui semble normal puisqu'au départ on a majoré les erreurs sur les variables. Mais cette supériorité dans les résultats ne semble pas obliger l'algorithme à considérer certains nombres nuls même s'ils sont très petits (ex: déterminant).

## 5. CONCLUSION

Ce chapitre a mis en évidence le problème de l'arithmétique en virgule flottante et les difficultés de maîtriser les erreurs. Il existe des analyses qui permettent d'obtenir l'influence des erreurs de données sur les résultats, telle que l'analyse différentielle, mais elles ne sont pas suffisante puisqu'elles ne tiennent pas compte des erreurs commises par l'ordinateur qui doivent s'y ajouter.

Nous envisageons d'utiliser les résultats obtenus à l'aide de l'arithmétique d'intervalles pour le calcul d'intersection de segments (ou de droites). Comme le déterminant du système dépend

de l'angle entre les deux droites $(\det((AB),(CD)) = \sin\alpha \times \left\|\vec{AB}\right\| \left\|\vec{CD}\right\|$ où $\alpha$ est l'angle entre les droites (AB) et (CD)), nous pouvons suivant l'ordre de grandeur du déterminant, estimer quelle sera l'erreur affectant les résultats. On pourra ainsi effectuer des rotations d'angles petits sans que soient considérés égaux l'objet initial et l'objet résultat.

Pour prendre en compte les erreurs commises par l'ordinateur, nous présentons, au chapitre suivant, la méthode CESTAC (Contrôle et Estimation STochastique des Arrondis de Calculs) développée par J. Vignes [VIG 93].

# CHAPITRE III

# CONTRÔLE STOCHASTIQUE

# DES ARRONDIS DE CALCUL

## 1. INTRODUCTION

Nous allons présenter une approche stochastique de l'analyse de la propagation des erreurs d'arrondi en calcul scientifique et une méthode d'estimation des erreurs d'arrondi. La méthode CESTAC (**C**ontrôle **S**tochastique des **A**rrondis de **C**alcul) permet, pour tout algorithme fini, de faire une estimation quantitative de cette propagation d'erreurs d'arrondi, et d'estimer le nombre de chiffres décimaux significatifs de tout résultat fourni par l'ordinateur. Elle a été introduite et développée par J. Vignes [VIG 78] [VIG 87] [VIG 93].

Nous expliquons les différents aspects théoriques et pratiques de la méthode de permutation-perturbation.

## 2. PROPAGATION DES ERREURS D'ARRONDI DANS UN ALGORITHME

Considérons une procédure algébrique finie nécessitant l'exécution de n opérations arithmétiques ordonnées et qui fournit un résultat unique $r \in \mathbf{R}$. Lorsqu'on l'exécute sur un ordinateur, on obtient alors un résultat informatique $R \in \mathbf{F}$, entaché d'une erreur absolue due à la propagation des erreurs d'arrondi.

Avec les notations définies au paragraphe 3 du chapitre I, on a le théorème suivant :

**Théorème 1 : [VIG 93]**

*Si R est le résultat informatique d'une procédure informatique P finie, ne faisant intervenir que les quatre opérations élémentaires, et, si r représente le résultat obtenu en effectuant la même suite de calculs mais avec une précision infinie, alors en ne retenant que les termes du premier ordre en $2^{-t}$ :*

$$R = r + \sum_{i=1}^{s_n} g_i(d)\, 2^{E_i - t}\, \delta_i \alpha_i + O(2^{-2t})$$

*$g_i$ (d) représente des constantes ne dépendant que des données et de l'algorithme,*

*$E_i$ les exposants des résultats intermédiaires,*

*$\alpha_i$ les quantités perdues lors de la troncature ou de l'arrondi*

*$\delta_i$ les signes des résultats intermédiaires,*

*n le nombre d'opérations élémentaires effectuées par l'ordinateur et d'affectation des données.*

La sommation va de 1 à $s_n$, car le nombre de termes intervenants dans la procédure "dépend de n", mais peut ne pas lui être égale.

On démontre ce théorème par récurrence en annexe 8.

# 3. CONTROLE STOCHASTIQUE DES ERREURS D'ARRONDI

## 3.1. Introduction

Étudier la propagation des erreurs d'arrondi, c'est étudier le comportement des $\alpha_i$. Les approches déterministes de ce problème sont pessimistes, car elles ne tiennent pas compte du côté compensatoire de la propagation des ces erreurs : on peut espérer que certaines de ces erreurs d'arrondi vont se compenser.

L'idée de l'approche stochastique est que, dans la mesure où on ne maîtrise pas les quantités $\alpha_i$ perdues à chaque étape, toutes les valeurs de $\alpha$, dans leur intervalle de variation, sont acceptables pour décrire le phénomène. Le principe de la méthode consiste donc, à interpréter les $\alpha_i$ non plus comme des quantités déterministes mais comme des quantités aléatoires.

Mais, pour une étude statistique, il est indispensable de faire plusieurs tirages de la population, or, par définition, les $\alpha_i$ sont des quantités qui disparaissent lors du calcul informatique. Il faut donc trouver une méthode qui permette d'étudier le plus fidèlement possible les $\alpha_i$ tout en introduisant un minimum d'erreurs supplémentaires et qui ne nécessite pas une longueur de mantisse supérieure.

M. La Porte et J. Vignes [VIG 84] ont développé une telle méthode appelée *méthode de Permutation-Perturbation* connue aussi sous le nom de CESTAC (**C**ontrôle **ST**ochastique des **A**rrondis de **C**alculs).

On présente les aspects théoriques et pratiques de cette méthode, et nous l'appliquons aux problèmes d'intersection de droites.

## 3.2. La méthode de permutation-perturbation ou CESTAC

### *3.2.1. Méthode de permutation-perturbation : Aspect théorique*

Considèrons une méthode numérique finie, définie par :

$$\pi = \text{procédure}(d, r, +, -, \times, /)$$

d étant l'ensemble des données, $d \subset \mathbf{R}$;

r étant l'ensemble des résultats cherchés, $r \subset \mathbf{R}$.

On admet que le résultat r est unique pour simplifier l'analyse, $r \in \mathbf{R}$.

Mise en oeuvre sur un ordinateur, cette procédure s'écrit sous forme syntaxique identique à la forme algébrique :

$$P = \text{PROCEDURE}(D, R, \underline{+}, \underline{-}, \underline{\times}, \underline{\div})$$

$D \subset \mathbf{F}$, $\mathbf{R} \subset \mathbf{F}$, $\mathbf{F}$ étant l'ensemble des flottants représentables en machine.

On ne peut pas considérer cette procédure informatique P comme l'image en machine de la procédure algébrique $\pi$, mais comme l'une des images. En effet, les règles de l'algèbre ne sont

plus vérifiées (cf Chap. I), chaque procédure informatique $P_i$ correspondant à une permutation possible des opérateurs permutables est aussi représentative de la procédure algébrique $\pi$.

• La méthode des permutations consiste donc à créer différents éléments de **R** en faisant exécuter par l'ordinateur les différentes procédures $P_i$, images de la procédure $\pi$.

De même, si on fait exécuter par la machine une procédure informatique $P_i$, image de la procédure algébrique $\pi$, chaque opérateur entraîne une erreur d'arrondi. Le résultat obtenu, à l'aide de cet opérateur, est toujours entaché d'une erreur. On doit considérer que toute opération informatique admet deux résultats : l'un par défaut, l'autre par excès, les deux résultats étant légitimes.

Si $x \in \mathbf{R}$ est le résultat exact d'une opération arithmétique, les résultats informatiques extrêmes représentant aussi légitimement x sont $\nabla x \in \mathbf{F}$ et $\Delta x \in \mathbf{F}$ et l'on a toujours $\nabla x$ et $\Delta x$ qui encadrent au plus près x :

$$\nabla x \in \leq x \leq \Delta x$$

($\nabla x$ est l'arrondi par défaut de x et $\Delta x$ est l'arrondi par excès)

• La méthode des perturbations consiste donc, pour chaque opération et affectation de chaque procédure $P_i$, à obtenir les valeurs par défaut et par excès des résultats.
On obtient ainsi un échantillon de valeurs représentant aussi équitablement le bon résultat.

• La méthode permutation-perturbation consiste à appliquer la méthode de perturbation à chacune des procédures informatiques images fournies par la méthode de permutation. On obtient alors l'ensemble $\underline{\underline{R}}$ des résultats informatiques images du résultat unique r d'un algorithme algébrique $\pi$ donné.

### *3.2.2. Aspect pratique de la méthode CESTAC*

Dans la pratique, il n'est pas possible de réaliser la totalité des éléments de $\underline{\underline{R}}$. La méthode pratique consiste à créer, en utilisant un logiciel approprié, N éléments de $\underline{\underline{R}}$ en faisant exécuter N fois une procédure $P_i$ en perturbant aléatoirement le dernier bit de la mantisse du résultat de chaque opération élémentaire : affectation, opérations arithmétiques, calcul de fonction bibliothèque. Dans la pratique, on prend N = 2 ou 3.
La perturbation aléatoire consiste à ajouter aléatoirement au dernier bit de la mantisse :

- la valeur 0 ou 1 avec une probabilité p telle que p(0) = p(1) = 0.5, si l'ordinateur travaille en arithmétique tronquée,
- la valeur −1 ou 0, ou +1 avec un probabilité p telle que p(−1) = p(+1) = 0.25 et p(0) = 0.5, si l'ordinateur travaille en arithmétique arrondie au plus près.

Ceci revient à considérer comme résultat informatique aléatoire, les valeurs du successeur $X^+$ ou du prédécesseur $X^-$ de la valeur X fournie habituellement par l'ordinateur :

$$X^+ = X + b^{E-t}$$
$$X^- = X - b^{E-t}$$

où E est l'exposant de X.

*3.2.3. Estimation de la précision d'un résultat informatique*

Pour estimer la précision d'un résultat informatique, on évalue le nombre de chiffres décimaux (quelle que soit la base utilisée par l'ordinateur, les résultats sont restitués dans la base décimale) significatifs de ce résultat.

Si $R_i$ est un élément quelconque de $\underline{\underline{R}}$ et R le résultat mathématique exact, l'évaluation statistique de l'erreur commise sur $R_i$, $R_i$ - R, peut être exprimée par :

$$\varepsilon_i = \sqrt{(R_i - \overline{R})^2 + \sigma^2}$$

où $\overline{R}$ est la moyenne des éléments de la population R et $\sigma^2$ la variance de cette population, et on a :

$C_i = \log_{10} \frac{|R_i|}{\varepsilon_i}$ le nombre de chiffres significatifs de l'éléments $R_i$.

Maillé [MAI 82] a montré que le meilleur estimateur de $R_i$ est donné par la valeur moyenne du caractère $\underline{\underline{R}}$ étudié et que les résultats $R_i$ peuvent être considérés comme une variable aléatoire Gaussienne (ou normale) de moyenne m et d'écart-type $\sigma$. Ils suivent donc une loi normale $N(m, \sigma)$.

Ici, nous n'avons qu'un échantillon aléatoire $(R_1, ..., R_n)$ de variables aléatoires indépendantes de loi identiques. Si $\overline{R}$ et $S^2$ sont respectivement la moyenne et la variance de l'échantillon aléatoire $(R_1, ..., R_n)$, alors on a $\overline{R}$ qui suit une loi normale $N\left(m, \frac{\sigma}{\sqrt{m}}\right)$ et $\sqrt{n-1}\left(\frac{\overline{R} - m}{S}\right)$ qui suit une loi du Khi-deux, $\chi^2_{n-1}$ à n−1 degré de liberté. De plus, les variables $\overline{R}$ et $S^2$ sont indépendantes.

Alors, la variable aléatoire $\sqrt{n-1}\left(\frac{\overline{R} - m}{S}\right)$ est de loi de Student à n−1 degré de liberté.

A l'aide de ces résultats, on peut déterminer un intervalle de confiance de la moyenne m de l'échantillon aléatoire.

$$\boxed{\overline{R} - \frac{t_{\beta/2} s}{\sqrt{n}} < m < \overline{R} + \frac{t_{\beta/2} s}{\sqrt{n}}}$$

où $t_{\beta/2}$ est déterminé à l'aide de la table de Student en fonction de $\beta$ et du degré de liberté n−1,

$$\overline{R} = \frac{1}{n}\sum_{i=1}^{n} R_i \text{ et } s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(R_i - \overline{R}\right)^2.$$

On obtient donc une erreur, sur l'approximation de la moyenne, plus petite que $2\frac{t_{\beta/2}s}{\sqrt{n}}$. A partir de cet intervalle de confiance, on peut alors définir le nombre de chiffres décimaux significatifs C de n'importe quel élément $R_i$ par la relation :

$$\boxed{C = \log_{10}\frac{\sqrt{n}\,\overline{R}}{2s.t_{\beta/2}}}$$

Toutefois, comme il est montré dans [CHE 88], si on considère la moyenne $\overline{R}$ à la place de l'un des tirages $R_i$ du résultat informatique, on gagne un facteur 2 et on obtient le nombre de chiffres significatifs $C_{\overline{R}}$ de $\overline{R}$ défini par :

$$\boxed{C_{\overline{R}} = \log_{10}\frac{\sqrt{n}\,\overline{R}}{s.t_{\beta/2}}}$$

On a donc les deux formules suivantes :

$$C_R = \log_{10}\frac{|\overline{R}|}{s} - K_1$$

et

$$\boxed{C_{\overline{R}} = \log_{10}\frac{|\overline{R}|}{s} - K_2}$$

avec $K_1 = \log_{10}\frac{2s.t_{\beta/2}}{\sqrt{n}}$ et $K_2 = \log_{10}\frac{s.t_{\beta/2}}{\sqrt{n}}$.

Ainsi, avec une probabilité de 95%, on obtient pour n = 3 : $K_1 = 0{,}69$ et $K_2 = 0{,}39$ et pour n = 2 : $K_1 = 1{,}25$ et $K_2 = 0{,}95$.

Remarque :

Si l'on s'intéresse au nombre de bits significatifs exacts, on prend $\log_2$ au lieu de $\log_{10}$, et dans ce cas, on gagne 1 bit significatif.

Les algorithmes de base de la méthode CESTAC sont décrits en annexe 9.

### *3.2.4. Validité, stabilité et robustesse de la méthode*

Ces problèmes ont été largement traités dans la littérature et nous nous contentons de les résumer et les citer.

Jean-Marie Chesnaux [CHE 88] a montré, en se basant sur une modélisation probabiliste de la méthode CESTAC, et sous des hypothèses vérifiées dans les problèmes réels, l'efficacité de la méthode pour estimer la précision des résultats de calculs sur ordinateurs. L'étude théorique qu'il a effectuée montre que l'erreur commise sur $1-\beta$ est au maximum, dans le cas de la troncature, de l'ordre de 0,0076 et de 0,048 pour l'arrondi. La probabilité de l'erreur sur l'intervalle de confiance est donc, au pire, de 5,8% pour la troncature (au lieu de 5% théorique) et de 5,5% dans le cas de l'arrondi. Il montre aussi que la méthode est robuste, c'est-à-dire, qu'elle fournit des estimations du nombre de chiffres significatifs exacts des résultats informatiques, même dans le cas où les hypothèses nécessaires à sa validité ne sont pas rigoureusement satisfaites. Une étude similaire sur le taux de fiabilité général de la méthode a aussi été effectuée par Françoise Chatelin [CHA 88]. On peut aussi consulter, pour plus de précision, le livre de M. Pichat et J. Vignes consacré à cette méthode [VIG 93].

### 3.3. Influence des erreurs de données sur les résultats d'algorithmes finis

Dans de nombreuses applications, les données sont connues avec une certaine incertitude (données issues de mesure etc...).

Il faut donc étudier l'influence de ces erreurs sur le résultat final, sans oublier pour autant les erreurs d'arrondi.

Pour conserver l'approche probabiliste [CHE 86], on considère les erreurs sur les données comme l'écart type des erreurs commises lors de la mesure; elles seront désormais assimilées à des variables aléatoires centrées (ces erreurs dues au hasard sont tantôt positives, tantôt négatives), et on admet qu'elles suivent approximativement une loi de Gauss.

Afin de tenir compte de ces erreurs avec la méthode de CESTAC, il suffit, lors des N exécutions du programme de faire sur chacune des données d, entachée d'une erreur relative $\varepsilon$, qui est l'incertitude sur la mesure de la donnée, une perturbation aléatoire définie par :

$$D = d(1 + \theta.\varepsilon)$$

avec :

- d est la donnée mesurée,
- $\varepsilon$ est l'erreur relative sur cette donnée,
- $\theta$ est une variable aléatoire répartie sur $[-1, +1]$,
- D est la valeur qui est utilisée à chaque exécution du programme et qui change lors de ces diverses exécutions.

A l'issu des N exécutions du programme, on obtient pour chaque résultat fourni, N valeurs et on en déduit le nombre de chiffres significatifs sur la moyenne des N valeurs, considérée comme le résultat informatique.

Ainsi, dans le cas où les données utilisées dans le programme sont entachées d'une erreur de données, on tient compte, à la fois de l'influence des erreurs de données et des erreurs d'arrondi, pour évaluer la précision de tout résultat informatique.

### 3.4. Zéro informatique

En analyse numérique, le zéro mathématique est un outil puissant pour contrôler la validité de la solution fournie par un algorithme numérique. Mais, en informatique, il en est tout autrement, et il est nécessaire d'élaborer un nouveau concept, celui du **zéro informatique** [VIG 86] [VIG 87] pour contrôler la validité de la solution d'un algorithme numérique exécuté sur un ordinateur.

Il est inconcevable d'espérer obtenir un zéro exact à partir de calculs effectués par l'ordinateur à cause de la propagation des erreurs d'arrondi. On est donc amené à introduire le concept de **zéro informatique** :

Soit $R \in \mathbf{F}$ la valeur du résultat d'un calcul sur ordinateur :

— si $R = 0$ en étant significatif, ou bien

— si R est quelconque mais non significatif (il ne représente qu'un effet cumulé d'erreurs),

alors **R** doit être considéré comme un **zéro informatique**, noté $\underline{0}$.

Par rapport au concept mathématique classique de "nullité", on ajoute ici celui de "sans signification". Une valeur $R \in \mathbf{F}$ résultant d'un calcul sur ordinateur est non significative, si elle ne représente qu'un effet cumulé d'erreurs d'arrondi. Cela veut dire que l'erreur absolue $\varepsilon$ qui entache la valeur est du même ordre de grandeur que la valeur elle-même.

En effet, le nombre de chiffres significatifs $C_R$ de R est donné par :

$$C_R = \log \frac{|R|}{\varepsilon}$$

si $\varepsilon \geq |R|$ alors $C_R \leq 0$, R est bien dépourvue de sens puisqu'elle ne possède pas un seul chiffre significatif.

Si $\varepsilon = 0$, alors $C_R = \infty$, autrement dit, R est connu avec une précision infinie, soit, en informatique, une précision égale au nombre maximal de chiffres décimaux codables dans la mantisse.

### 3.5. Implémentation asynchrone de la méthode

Cette implémentation consiste à faire exécuter successivement N fois (2 ou 3) le programme de calcul en utilisant les logiciels d'arithmétique aléatoire de la méthode CESTAC. On conserve les résultats obtenu dans un tableau à N dimensions, afin de pouvoir en déterminer le nombre de chiffres significatifs à la fin des N exécutions.

L'implémentation asynchrone de la méthode CESTAC permet d'avertir l'utilisateur si le résultat fourni est significatif ou non ([VIG 93]), et de mettre en évidence les cas où le programme ne peut pas être correctement exécuté sur ordinateur.

Elle évite donc à la machine de fournir un résultat faux, mais elle ne permet pas de déterminer d'où vient l'impossibilité de fournir un résultat exact, ni de pouvoir obtenir le résultat exact.

De tels problèmes sont souvent causés par les instructions conditionnelles des programmes. Une autre implémentation de la méthode permet de les résoudre, mais avant de la détailler, nous exposons les problèmes des instructions conditionnelles.

### 3.6. Le problème des instructions conditionnelles

Les algorithmes contiennent souvent des instructions conditionnelles qui se présentent entre autre sous la forme :

si $(a \geq b)$ alors instruction 1 sinon instruction 2.

Ils peuvent toujours s'écrire :

si $((a - b) \geq 0)$ alors instruction 1 sinon instruction 2.

$a \in \mathbf{R}$ et $b \in \mathbf{R}$ étant des résultats d'expressions arithmétiques.

Dans le programme de calcul de l'ordinateur, on a l'instruction conditionnelle suivante :

si $((A \ominus B) \geq 0)$ alors instruction 1 sinon instruction 2.

$A \in \mathbf{F}$ et $B \in \mathbf{F}$ étant des résultats d'expressions arithmétiques calculées en virgule flottante, donc toujours entachés d'erreurs d'arrondi.

Lorsque ces expressions sont calculées N fois en utilisant la méthode CESTAC, on obtient N valeurs $A_i$ et $B_i$ ainsi que leur moyenne $\overline{A}$, $\overline{B}$ et leur nombre de chiffres décimaux significatifs $C_{\overline{A}}$, $C_{\overline{B}}$.

Trois cas peuvent se présenter :

1. Les valeurs de $\overline{A}$ ou $\overline{B}$ et $\overline{A_i \ominus B_i}$ sont significatives. Dans ce cas, les N valeurs $A_i \ominus B_i$ ont le même signe et à chaque exécution, c'est la même séquence de calcul qui est exécutée. Il n'y a donc pas de problème.

2. Les valeurs de $\overline{A}$ ou $\overline{B}$ sont significatives, mais la valeur de $\overline{A_i \ominus B_i}$ ne l'est pas. Dans ce cas, lors des N exécutions, les N valeurs $A_i \ominus B_i$ peuvent ne pas avoir le même signe, et alors les instructions 1 et 2 peuvent être aléatoirement exécutées. De ce fait, le nombre de chiffres décimaux significatifs du résultat ne peut pas être calculé. Il y a donc un problème. On doit en avertir l'utilisateur afin qu'il puisse prendre un décision. Néanmoins dans ce cas, au sens

informatique, $\overline{A} = \overline{B}$ (ce qui ne veut pas dire a = b), il est alors logique que l'instruction correspondant à l'égalité soit exécutée, en précisant que l'on a un zéro informatique.

3. Les valeurs de $\overline{A}$ et $\overline{B}$ sont non significatives. C'est le cas d'indécidabilité, le débranchement ne peut pas être interprété, il ne doit pas être exécutée.

Pour pouvoir résoudre ces problèmes, il faut pouvoir impérativement détecter la présence de cette valeur non significative ce qu'il n'est pas possible de faire avec l'implémentation asynchrone de la méthode CESTAC. Il faut implémenter la méthode de façon synchrone.

### 3.7. Implémentation synchrone

Cette implémentation (détaillée dans [FAY 89]) consiste à effectuer N fois chaque opération à virgule flottante, en utilisant l'arithmétique aléatoire, avant d'effectuer la suivante dans le programme. Tout se passe comme si N programmes identiques se déroulaient en même temps sur N ordinateurs synchronisés utilisant chacun une arithmétique aléatoire. Ainsi, il est possible d'estimer la précision, c'est à dire le nombre de chiffres décimaux significatifs de tout résultat intermédiaire du programme et par conséquent de détecter toute valeur non significative (zéro informatique).

L'utilisateur peut en être averti et en faire l'interprétation qu'il veut. En effet, le zéro informatique peut être, ou non, considéré comme un zéro mathématique suivant les cas. Pour cela une analyse du programme est nécessaire.

## 4. EXEMPLES

**Exemple 1 :**

Nous présentons ici une comparaison entre le modeleur Sacado et le calcul d'intersection avec l'implémentation de la méthode CESTAC. Cette étude est effectuée en fonction de l'angle formé par les deux droites, et nous présentons que les erreurs maximales, ainsi que l'estimation du nombres de chiffres significatifs correspondants (Tableau 7).

Tableau 7 : Erreurs commises avec la méthode CESTAC

| Angle | Erreurs CESTAC | Chiffres significatifs | Erreurs Sacado |
|---|---|---|---|
| 1 | 9,8E-12 | 11,7 | 1,2E-13 |
| 0,1 | 2E-11 | 10,2 | 2E-12 |
| 0,01 | 5E-10 | 9,6 | 4E-11 |
| 0,001 | 9E-09 | 8 | det = 0 |
| 0,0001 | 9,8E-08 | 6,7 | det = 0 |

Dans cet exemple, nous voyons que l'estimation du nombre de chiffres significatifs, correspond à un chiffre près, au nombre de chiffres réellement significatifs des coordonnées déterminées par la méthode CESTAC. Par rapport au logiciel Sacado, les valeurs sont un peu moins significatives, car on a volontairement introduit des erreurs. Par contre, le test sur le déterminant, pour savoir s'il est nul ou non, est meilleur que celui de Sacado. En effet, nous estimons que le déterminant est nul s'il ne correspond qu'à l'accumulation des erreurs, c'est à dire, si nous avons un zéro informatique. Ceci a était possible, car nous avons effectué une implémentation synchrone de la méthode.

**Exmple 2**

Nous reprenons ici l'exmple du pentagone décrit au chapitre I.

Tableau 8 : Erreurs et estimation du nombre de chiffres significatifs pour l'opération $out^n(in^n(A))$

| p | $out^1in^1(A))$ | | | $out^2(in^2(A))$ | | | $out^3(in^3(A))$ | | | $out^4(in^4(A))$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ER | E | CS | ER | E | CS | ER | E | CS | ER | E | CS |
| 0,1 | 1E-15 | 1E-15 | 14,3 | 6E-15 | 4E-15 | 12,6 | 2E-13 | 7E-12 | 10,2 | 4E-13 | 7E-10 | 8 |
| 0,01 | 0 | 5E-15 | 14,2 | 6E-12 | 3E-10 | 10,8 | 9E-11 | 4E-08 | 6,1 | 5E-08 | 3E-04 | 4,4 |
| 0,001 | 1E-13 | 2E-13 | 12,5 | 4E-10 | 1E-09 | 9,5 | 4E-07 | 8E-07 | 7,7 | det=0 | det=0 | |
| 0,0001 | 1E-13 | 3E-13 | 12,1 | 2E-10 | 3E-08 | 7,8 | det=0 | det=0 | | det=0 | det=0 | |

Tableau 9 : Erreurs et estimation du nombre de chiffres significatifs pour l'opération $in^n(out^n(A))$

| p | $in^1(out^1(A))$ | | | $in^2(out^2(A))$ | | | $in^3(out^3(A))$ | | | $in^4(out^4(A))$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ER | E | CS | ER | E | CS | ER | E | CS | ER | E | CS |
| 0,1 | 0 | 1E-16 | 14,3 | 2E-15 | 5E-15 | 12,6 | 2E-13 | 8E-12 | 10,2 | 6E-12 | 7E-10 | 8 |
| 0,01 | 0 | 5E-15 | 14,2 | 1E-14 | 3E-14 | 13.0 | 5E-11 | 5E-12 | 10 | 8E-08 | 3E-04 | 5,6 |
| 0,001 | 0 | 2E-14 | 13,6 | 1E-10 | 5E-09 | 7,5 | 9E-08 | 3E-07 | 8 | 2E-01 | det=0 | |
| 0,0001 | 0 | 2E-13 | 12,9 | 1E-08 | 7E-06 | 5,8 | 7E-05 | det=0 | 0 | 8E00 | det=0 | |

Nous avons remis ici les erreurs commises par l'ordinateur (c'est à dire en arithmétique flottante double précision), afin que les comparaisons soient plus visibles. Nous voyons que le fait d'introduire volontairement des erreurs influent considérablement sur les résultats. Ceci est d'autant plus vrai dans cet exemple, car plus n est grand, plus on effectue de perturbations. Cependant, nous avons toujours à notre disposition la connaissance de la précision. Ici ,comme

dans l'exemple précedent, le déterminant est nul dès que qu'il ne représente plus que l'effet cumulé des erreurs.

## 5. CONCLUSION

Nous venons de présenter la méthode CESTAC, qui permet de déterminer le nombre de chiffres significatifs d'un résultat obtenu par une suite de calculs en arithmétique flottante. Nous voyons que les techniques d'implémentation de la méthode doivent être adaptées à l'algorithme de la procédure à perturber. Dès qu'une instruction conditionnelle est utilisée dans une procédure, il faut en toute rigueur utiliser une implémentation synchrone, car les résultats de calcul risquent de ne plus être comparables. Plus généralement, l'implémentation synchrone de CESTAC permet d'une part, de gérer correctement les instructions conditionnelles dans toute procédure informatique, et d'autre part, d'estimer la précision de tout résultat intermédiaire.

Nous remarquons aussi que cette méthode introduit volontairement des erreurs, qui parfois deviennent grandes. Même si elles ont le même comportement que les erreurs introduites par l'ordinateur, il faut donc pouvoir éviter ce désagrément. C'est ce que nous proposons de faire en ayant recours à l'arithmétique exacte dans les cas critiques.

En intégrant cette méthode au logiciel Sacado, nous pourrons déterminer les cas où une arithmétique exacte s'avère nécessaire. La méthode CESTAC ne permettant pas de corriger les erreurs, elle est insuffisante pour répondre aux problèmes de précisions numériques auxquels nous sommes confrontés si elle est utilisée seule.

Dans le chapitre suivant, nous effectuons une étude de l'arithmétique exacte que nous voulons utiliser en ayant recours à un logiciel de calcul formel.

# CHAPITRE IV

# CALCUL FORMEL

## 1. INTRODUCTION

Parmi les solutions possibles au problème des imprécisions numériques, l'utilisation d'une arithmétique exacte est la plus simple et la plus exacte pour les concepteurs d'algorithmes. Ils peuvent être implantés tels quels sans se préoccuper des erreurs numériques. Malheureusement, les implications en temps et en mémoire des arithmétiques exactes découragent souvent. Il convient donc de n'utiliser cette arithmétique que dans les cas critiques, cet aspect sera développé au chapitre V de cette thèse. Nous présentons dans ce chapitre un moyen pour éviter le problème introduit par l'utilisation de grands entiers et l'arithmétique spéciale nécessaire, le calcul formel.

Dans la section 2, nous présentons les avantages et inconvénients du calcul formel qui nous ont conduit à utiliser le logiciel Pari [BCO 95] [SHA 93].

En section 3, nous présentons ce logiciel et le moyen d'effectuer l'interface entre Sacado et Pari. Dans la section 4, nous abordons le problème de l'approximation d'un nombre flottant par un rationnel et nous présentons quatre méthodes résolvant cette question.

Nous terminons ce chapitre par différents tests qui montrent l'apport d'une arithmétique exacte.

## 2. ASPECT GÉNÉRAL D'UN SYSTÈME DE CALCUL FORMEL

### 2.1. Généralités

Avec l'apparition des ordinateurs, le "calcul numérique" est devenu prépondérant par rapport au "calcul algébrique". Néanmoins, le premier n'élimine pas le second : l'écriture du moindre programme numérique nécessite la mise au point des formules sur lesquelles est basé l'algorithme. Pendant longtemps, et encore aujourd'hui, beaucoup de ces calculs sont effectués à la main. Un temps considérable est souvent nécessaire pour aboutir aux formules finales. Un des calculs les plus impressionnants est celui que fit Delaunay au XIX$^{ème}$ siècle pour calculer l'orbite de la Lune. Il mit une dizaine d'année pour la calculer. Le résultat final est une formule non numérique qui occupe 128 pages. Aujourd'hui, avec un système sachant effectuer des opérations, telles que les développements limités, les dérivations, intégrations, on obtient en quelques minutes le même résultat que Delaunay.

Les principaux types utilisés sont les entiers, les rationnels, les réels et les complexes en précision illimitée. On y trouve également les polynômes, les fractions rationnelles, les matrices à éléments numériques et/ou symboliques, suites, séries etc... . Ces types sont accompagnés des opérations habituelles, ainsi que la dérivation, l'intégration, le développement en série, les

diverses manipulations de formules. Tout ceci ouvre la voix à différents champs d'application : la résolution d'équation, l'intégration formelle, le calcul des limites etc ... [GOM 95] [DAV 93].

Bien souvent, un calcul nécessite l'utilisation de plusieurs instructions, c'est pourquoi la plupart des systèmes de calcul formel sont dotés d'un langage de programmation qui leur est propre.

Après cette présentation succincte, nous analysons les avantages et les défauts du calcul algébrique sur ordinateur, en particulier, en ce qui concerne le problème d'imprécision numérique lié au logiciel Sacado.

### 2.2. Avantages

• *Manipulation des formules*

Dans les calculs longs et fastidieux, il apporte **la vitesse et la sûreté** par rapport aux calculs effectués à la main. De plus, les formules peuvent être simplifiées plus facilement avec le logiciel qu'en effectuant les calculs à la main. Par exemple, pour le calcul d'erreur de polygonalisation $\varepsilon_p$ d'un arc de cercle de centre (x,y), de rayon r, variant de l'angle a à l'angle b [GAP 95], nous obtenons : $\varepsilon_p = \sqrt{(x_{ci} - x_i)^2 + (y_{ci} - y_i)^2}$ où $(x_i,y_i)$ sont les coordonnées du milieu d'un segment de polygonalisation et $(x_{ci},y_{ci})$, ceux du point du cercle correspondant. Ici, la formule paraît simple, mais elle cache les calculs des coordonnées précédemment citées. En les remplaçant par les formules contenant les données du problème et en se plaçant au premier segment, on obtient :

$$\varepsilon_p = r \times \left[ \left( \cos\left(a + \frac{b-a}{2n}\right) - \frac{1}{2}\cos(a) - \frac{1}{2}\cos\left(a + \frac{b-a}{n}\right)\right)^2 + \left( \sin\left(a + \frac{b-a}{2n}\right) - \frac{1}{2}\sin(a) - \frac{1}{2}\sin\left(a + \frac{b-a}{n}\right)\right)^2 \right]^{\frac{1}{2}}$$

Simplifier cette formule n'est pas sans risque de commettre des erreurs. En utilisant le logiciel Maple, en quelques minutes, on aboutit au résultat suivant :

$$\varepsilon_p = r \times \left[ \frac{3}{2} + \frac{1}{2}\cos\left(\frac{a-b}{n}\right) - 2\cos\left(\frac{a-b}{2n}\right)\right]^{\frac{1}{2}}$$

On constate qu'il y a moins de calculs pour le même résultat. Nous aurons donc un gain de temps à l'exécution et une propagation d'erreurs moins importante.

• *Précision*

La taille des nombres flottants est fixée par l'utilisateur et peut être aussi grande que l'on veut. C'est ce que l'on appelle *la précision arbitraire*.

Ceci permet d'obtenir **une grande précision** dans les résultats, car, même si les données et les résultats sont de tailles raisonnables, les calculs intermédiaires peuvent engendrer des nombres de très grande taille.
De plus, si on utilise l'arithmétique rationnelle, on élimine toutes les imprécisions numériques.

### 2.3. Inconvénients

• *Le temps de calcul*

La principale faiblesse des systèmes de calcul formel est leur lenteur pour le calcul numérique. Elle est due aux structures de données qui sont plus générales que celles des systèmes numériques, et qui sont donc lourdes à manipuler. De plus, la compilation de l'arithmétique flottante n'est pas efficace dans ces systèmes. Lorsque ces instructions sont élémentaires comme l'addition de deux nombres flottants, la compilation diminue le temps de calcul d'un facteur pouvant aller jusqu'à 100.

• *Lien avec d'autres langages*

En raison de la lenteur de ces systèmes, le recours à des langages tels que C ou Fortran est indispensable. En effet, lorsqu'une petite boucle numérique utilise la majeure partie du temps de calcul, il peut s'avérer fructueux d'écrire cette boucle dans un langage compilé tel que C ou Fortran.

Certains logiciels de calcul formel ont des bibliothèques, de manière à pouvoir les utiliser à partir de langages tels que Fortran ou C. C'est le cas, en particulier, de Matlab [MAT 95] et de Pari [BCO 95]. Bien que le langage de programmation de certains logiciels de calcul formel (Malpe [MAP 93], Matlab [MAT 95]), permet d'avoir recours aux langages compilés, il semble plus aisé d'avoir recours à un des logiciels de calcul formel à partir de modeleur Sacado, que d'effectuer la démarche inverse.

### 2.4. Conclusion

Les systèmes de calcul formel sont très efficaces et fiables pour la manipulation des formules. Ils devraient être utilisés dans les laboratoires de recherche scientifique pour l'élaboration d'expressions mathématiques. Ils permettent d'être plus rapide et d'éviter les erreurs humaines.

En ce qui concerne le calcul numérique, ils sont lents, surtout si l'on veut une grande précision dans les calculs. Pour exploiter les avantages d'un tel système, il faudrait créer une interface entre lui et le logiciel Sacado.

Le problème auquel on est confronté dans le modeleur Sacado, est un problème de précision et de propagation des erreurs. Il faudrait pouvoir effectuer, dans un premier temps, certaines fonctions de base, avec une arithmétique exacte. Cette arithmétique est coûteuse en temps et en espace mémoire, car il faudrait la gérer soit même. En plus, les entiers deviennent

rapidement plus grands que la capacité de la machine. Divers ouvrages traitent spécialement du calcul formel : [GOM 95], [DAV93].

Il existe d'autres systèmes qui ne sont pas réellement des systèmes de calculs formels, mais qui offrent de nombreuses possibilités. Comme nous recherchons un logiciel assez rapide et offrant une interface simple, nous nous sommes intéressés au logiciel PARI [BCO 95], que l'on peut utiliser comme une bibliothèque de fonctions arithmétiques en C. Il semble être efficace et assez rapide. Il est donc intéressant de tester l'efficacité de l'utilisation de PARI dans Sacado.

## 3. LE SYSTÈME PARI

### 3.1. Présentation du système Pari

Le système Pari n'est pas réellement un système de calcul formel. Il fait parti des systèmes spécialisés dans certains domaines. Bien que beaucoup de manipulations symboliques soient possibles avec Pari, il est moins bon que des systèmes comme Axiom, Maple, Reduce, ou Mathematica, pour des domaines d'application tels que les polynômes à variables multiples, l'intégration formelle. Mais, il a le grand avantage d'être rapide. Il peut être de 5 à 100 fois plus rapide dans les calculs que les systèmes mentionnés précédemment [BCO 95].

On peut utiliser ce système de deux manières différentes :

- Comme un calculateur programmable, nommé GP
- Comme une bibliothèque de fonctions arithmétiques en C

Le deuxième moyen permet une interface aisée entre Pari et le modeleur Sacado, car on peut faire appel à des fonctions de Pari comme à des fonctions C quelconques.

Une caractéristique de Pari est la récursivité : presque tous les types employés sont récursifs. Par exemple, un nombre complexe peut avoir ses parties réelles et imaginaires de n'importe quel type (entiers, rationnels, réels ...). Cependant, il faut être prudent et ne pas créer par exemple des nombres complexes de nombres complexes. Pour plus de détails du système, on peut consulter le manuel d'utilisation [BCO 95].

### 3.2. Interface Pari - Modeleur

Pour utiliser la librairie de Pari, il faut écrire un programme en langage C, et le lier à la bibliothèque et aux fichiers inclus de Pari.

Il faut inclure le fichier contenant les outils disponibles sous Pari. Pour cela, il suffit d'inclure la bibliothèque par l'opération suivante:

```
#include <genpari.h>
```

Ce fichier apporte les constantes, variables et fonctions nécessaires et définit le type fondamental pour tous les objets de Pari : le type **GEN**, qui est tout simplement un pointeur sur le type "long".

Il faut ensuite initialiser le système, en particulier, l'espace mémoire qui contiendra tous les calculs. Il suffit simplement de faire appel à la fonction à deux variables **init**. Le premier argument est le nombre de bytes donnés à Pari pour travailler (pas moins de 500 000), le second est la limite supérieure de la table des nombres premiers pré-calculés. Si on ne veut pas de nombre premiers, on donne 2 comme argument.

A partir de ce moment, on peut utiliser aisement toutes les fonctions et variables disponibles. Il suffit de respecter les différentes instructions de programmation détaillées dans [BCO 95].

### 3.3. Conclusion

On a à présent à notre disposition une interface facile entre Sacado et un logiciel de calcul formel tel que Pari. On peut effectuer des opérations sur des rationnels sans se soucier de l'espace utilisé par les entiers et sans gérer soi-même de telles opérations.

On va maintenant mettre en pratique cette interface avec l'algorithme d'intersection de deux droites et effectuer des tests afin de savoir exactement ce que peut apporter, au logiciel Sacado, une telle interface.

Comme Sacado utilise des nombres flottants double précision, il nous faut étudier la conversion d'un nombre flottant en nombre rationnel et la précision avec laquelle cette approximation est faite, afin de pouvoir calculer l'intersection de deux droites à l'aide des rationnels de Pari.

## 4. ALIGNEMENT D'UN FLOTTANT SUR UN RATIONNEL

### 4.1. Introduction

On décrit quatre méthodes qui permettent d'approcher un nombre flottant par un nombre rationnel. On étudie en particulier, la convergence et la précision de la méthode de développement en fraction continue.

### 4.2. Décodage de la représentation binaire

Un nombre flottant peut être considéré comme un nombre "rationnel" dans le format binaire de la machine. Il est donc toujours possible de le transcrire dans un autre format choisi pour la représentation des rationnels.

- Avantages :

Cette méthode permet de convertir un flottant en rationnel sans aucune perte d'information.

- Inconvénients :

Elle est coûteuse et produit des rationnels inutilement encombrants.

### 4.3. Grille rationnelle

Les extrémités initiales (flottants) sont forcées sur les noeuds les plus proches d'une "grille entière régulière", dont le "pas" est fonction de la précision souhaitée. Cela revient à "arrondir" chaque coordonnée initiale sur le nombre entier le plus proche.

- Avantages :

Cette méthode produit toujours, au cours des calculs, des nombres rationnels bornés.

- Inconvénients :

Approcher un réel par entier entraîne une grande perte de précision et les nombres représentables sont limités par la borne de la grille.

### 4.4. Méthode naïve d'approximation

Soient w le réel à approcher et Q la borne maximale du dénominateur. Si $p = [wQ]$ est l'entier le plus proche de wQ, alors le rationnel p/Q est une approximation de w. De plus, comme $| wQ - [wQ] | \leq 0,5$, l'erreur d'approximation est bornée par 1/(2Q) [HOF 89].

Exemple :

Soit à approcher $w = 0,123$ avec $Q = 100$, on a $p = [0,123*100] = 12$ et 12/100 est une approximation de w. La perte de précision est alors de 0,003 ( $\leq 0,005$ ).

- Avantages :

Cette méthode construit en temps constant une approximation du nombre flottant et donne une borne de l'erreur commise.

- Inconvénients :

Les entiers du rationnel deviennent grands dès que l'on veut une bonne précision et de plus le rationnel n'est pas sous forme irréductible.

### 4.5. Développement en fractions continues

**Définition**

On définit la fonction de N+1 variables

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \quad \cdots \\ \qquad + \cfrac{1}{a_N}}} = [a_0, a_1, \ldots, a_N]$$

comme une **fraction continue finie**, ou, quand il n'y a pas d'ambiguïté, **fraction continue**.

Les $a_0$, $a_1$, ..., $a_N$ sont des réels avec $a_1$, ..., $a_N > 0$.

La fraction continue est appelée **simple** si les $a_0$, $a_1$, ..., $a_N$ sont des entiers.

On définit aussi les **fractions continiues infinies**, comme les limites de fractions continues finies.

**Théorème 2:** [ROS 93]

*Si $p_n$ et $q_n$ sont définis par :*

$$\begin{cases} p_0 = a_0 \\ p_1 = a_1 a_0 + 1 \\ p_n = a_n p_{n-1} + p_{n-2} \quad (2 \le n \le N) \end{cases} \quad et \quad \begin{cases} q_0 = 1 \\ q_1 = a_1 \\ q_n = a_n q_{n-1} + q_{n-2} \quad (2 \le n \le N) \end{cases}$$

*alors (Annexe 10)*

$[a_0, a_1, \ldots, a_n] = \frac{p_n}{q_n}$ *est la $n^{ième}$ réduite d'une fraction continue.*

**Théorème 3:** [HAW 60], [ROS 93], [LUC 61]

*Tout nombre irrationnel peut être exprimé d'une façon unique par une fraction continue simple infinie (Annexe 11).*

Remarque : Tout nombre rationnel peut être représenté par une fraction continue finie simple.

Pour calculer la fraction continue approchant le réel, on utilise l'algorithme d'Euclide :

**Algorithme d'Euclide :**

Soit x un réel, on note [x] la partie entière de x. On a alors :

$a_0 = [x]$ $\quad x = a_0 + \xi_0$ avec $0 \le \xi_0 \le 1$

Si $\xi_0 \neq 0$ alors on peut écrire :

$a'_1 = \frac{1}{\xi_0}$, $\quad [a'_1] = a_1$, $\quad a'_1 = a_1 + \xi_1$ $\quad$ avec $0 \le \xi_1 \le 1$

Si $\xi_1 \neq 0$ alors on peut écrire :

$$a'_2 = \frac{1}{\xi_1}, \quad [a'_2] = a_2, \quad a'_2 = a_2 + \xi_2 \quad \text{avec } 0 \le \xi_2 \le 1$$

Et ainsi de suite. Alors $a'_n = \frac{1}{\xi_{n-1}} > 1$, et ainsi $a_n \ge 1$ pour tout $n \ge 1$.

Ainsi $x = [a_0, a'_1] = \left[a_0, a_1 + \frac{1}{a'_2}\right] = [a_0, a_1, a'_2] = [a_0, a_1, a_2, a'_3] = \ldots$, où $a_0, a_1, a_2, \ldots$ sont entiers et $a_1, a_2, \ldots > 0$.

Le système d'équations :
$$\begin{cases} x = a_0 + \xi_0 & (0 \le \xi_0 \le 1), \\ \frac{1}{\xi_0} = a'_1 = a_1 + \xi_1 & (0 \le \xi_1 \le 1), \\ \frac{1}{\xi_1} = a'_2 = a_2 + \xi_2 & (0 \le \xi_2 \le 1), \\ \cdots & \end{cases}$$

est connue comme l'algorithme des fractions continues. Cet algorithme continue aussi longtemps que $\xi_n \neq 0$. Si on atteint une valeur de n, notée N, pour laquelle $\xi_N = 0$, alors l'algorithme termine et $x = [a_0, a_1, \ldots, a_N]$. Dans ce cas, x est représentée par une fraction continue simple et il est rationnel.

Si on utilise les fonctions $p_n$ et $q_n$ définies précédemment, on obtient l'algorithme suivant, avec en plus un test d'arrêt pour les nombres irrationnels :

entrée : réel à approcher x

sortie : deux entiers p et q tels que p/q approche x au mieux

```
début
  si x est un entier alors p ← [x] et q ← 1
  sinon {calcul de p et q}
    a0 ← [x]
    p0 ← a0 ; q0 ← 1 ; x1 ← 1/(x − a0) ; a1 ← [x1] ; p1 ← a1a0 + 1 ; q1 ← a1
    dif ← |( x − p1/q1)/x |
    tant que (dif > eps ) faire
      xn ← 1/(xn-1 − an-1)
      an ← [xn]
      pn ← anpn-1 + pn-2 ; qn ← anqn-1 + qn-2
      dif ← | (x − pn/qn)/x |
    ftq
    p ← pn
    q ← qn
  fsi
fin
```

**Précision**

*Les réduites, de la fraction continue simple infinie d'un nombre irrationnel $\alpha$, sont de bonnes approximation de $\alpha$..*

La preuve du théorème nous montre que, si $\frac{p_k}{q_k}$ est la $k^{ième}$ réduite de cette fraction continue, alors $\left|\alpha - \frac{p_k}{q_k}\right| < \frac{1}{q_k^2}$, et ceci quelque soit k.

• Avantages :

Cette méthode donne une bonne approximation du nombre flottant par des fractions *irréductibles*, ce qui évite d'avoir des numérateurs et dénominateurs plus grands que ce qu'il est nécessaire. De plus, on peut approcher le réel avec la précision que l'on veut.

• Inconvénients :

Le nombre d'opérations nécessaires est aléatoire et peut s'avérer grand. Si en plus, on veut une grande précision, les rationnels seront plus encombrants et les calculs sur les rationnels seront plus lents.

### 4.6. Conclusion

On voit que l'on peut passer d'un nombre de plusieurs manières. La méthode que nous avons choisie est la méthode des fractions continues, car elle crée des rationnels irréductibles, donc des entiers moins encombrants que les autres méthodes, et la perte de précision peut être maîtrisée. On peut se référer à [HAW 60], [ROS 93], [LUC 61].

## 5. ETUDES EXPERIMENTALES

### 5.1. Introduction

Nous proposons ici diverses études comparatives entre les calculs effectués par Sacado et ceux effectués à l'aide de rationnels en utilisant Pari, pour l'intersection de deux segments. On reprend les divers exemples du chapitre II.

Les calculs à l'aide de Pari se déroulont ainsi :

• On approche chaque coordonnées de chaque sommet par un rationnel.
• Chaque rationnel est transformé en un rationnel de Pari.
• On effectue le calcul du point d'intersection à l'aide de l'arithmétique de Pari.
• Les rationnels calculés (abscisse et ordonnée) sont transformés de nouveau en flottants.

### 5.2. Étude 1

Nous reprenons l'exemple de l'étude 1 du paragraphe 4.3. du chapitre II. Nous considérons le segment de sommets [-20,-20] et [30,30], auquel nous effectuons des rotations ayant pour centre le milieu du segment et des angles de plus en plus petits. Nous obtenons de nouveaux segments et nous calculons l'intersection entre chacun de ces segments et le segment initial avec le modèle Sacado et avec les rationnels de Pari. Le résultat théorique est bien entendu le milieu du segment, soit le point [5,5]. Les résultats pour le logiciel Sacado ne sont pas rappelés (voir chapitre II §4.2 Études expérimentales).

**Résultats pour les calculs effectués avec Pari :**

| ANGLE | INTERSECTION | EX | EY |
|---|---|---|---|
| (1E-02)° | oui | 0 | 0 |
| (1E-03)° | oui | 0 | 0 |
| (1E-04)° | oui | 0 | 0 |
| (1E-05)° | oui | 0 | 0 |
| (1E-06)° | oui | 0 | 0 |
| (1E-07)° | oui | 0 | 0 |

Ici, quelque soit l'angle de rotation, il n'y a pas eu d'erreur de calcul (au moins jusqu'à l'ordre 1E-14). Les erreurs de calculs sont inexistants puisqu'ils sont effectués de façon exacte avec une arithmétique rationnelle. Les seules erreurs existantes sont celles dues à l'approximation des flottants par des rationnels. Or, en ce qui concerne le segment initial, nous n'avons aucune erreur d'estimation puisque l'on a des entiers, il ne reste donc que les erreurs sur les sommets de chaque segment calculé par rotation, et ces dernières ne semble pas influencer les calculs.
Le calcul en arithmétique exacte apporte donc ici une plus grande précision, et permet des calculs d'intersection de segments corrects pour de petits angles.

### 5.3. Étude 2

Nous reprenons ici le problème du pentagone étudié au paragraphe II.3.4. Nous allons calculer $in^n(out^n(A))$ et $out^n(in^n(A))$ (le résultat théorique étant A) pour différentes valeurs de n et p et ceci de trois manières différentes.

1) Valeurs réelles

Nous avons effectué dans un premier temps les calculs avec les valeurs réelles en double précision (15 chiffres décimaux significatifs). Le déterminant est considéré nul si sa valeur est plus petite que 1E-10. Dans ce cas, on renvoie des valeurs fictives, mais le processus n'est pas

arrêté et on obtient alors de fausses valeurs dans les résultats que l'on ne doit pas considérer comme des erreurs de calculs. De tels résultats ne seront donc pas pris en compte, ainsi que les cas où certaines coordonnées de sommets ont pu être calculées et pas d'autres, car on peut considérer que le processus a échoué (case vide). Seule l'erreur absolue maximale de chaque cas est représentée.

Tableau 10 : Erreurs absolues maximales pour $out^n(in^n(A))$ en arithmétique flottante

| p | $out^1(in^1(A))$ | $out^2(in^2(A))$ | $out^3(in^3(A))$ | $out^4(in^4(A))$ |
|---|---|---|---|---|
| 0,1 | 1E-15 | 6E-15 | 2,25E-13 | 4,37E-13 |
| 0,01 | 0 | 5,92E-12 | 9,27E-11 | 5,12E-08 |
| 0,001 | 1,11E-13 | 4,21E-10 | 3,89E-07 | |
| 0,0001 | 1,11E-12 | 1,84E-10 | | |
| 0,00001 | | | | |

Tableau 11: Erreurs absolues maximales pour $in^n(out^n(A))$ en arithmétique flottante

| p | $in^1(out^1(A))$ | $in^2(out^2(A))$ | $in^3(out^3(A))$ | $in^4(out^4(A))$ |
|---|---|---|---|---|
| 0,1 | 0 | 2E-15 | 1,77E-13 | 5,96E-12 |
| 0,01 | 0 | 1,1E-14 | 5,18E-11 | 7,37E-08 |
| 0,001 | 0 | 1,033E-10 | 9,30E-08 | 2,10E-04 |
| 0,0001 | 0 | 1E-08 | 7,45E-05 | 8E+00 |
| 0,00001 | 0 | 2,35E-06 | 1,05E-01 | |

Que ce soit le tableau 10 ou le tableau 11, nous constatons comme au paragraphe II.3.4 que l'erreur augmente avec la profondeur de calcul n. Ceci s'explique par le nombre croissant d'opérations et donc une accumulation d'erreurs plus importantes. De même, quand p devient petit, l'angle entre certains segments est plus petit, et on a une perte plus importante d'informations qui se propage elle aussi.

2) Valeurs rationnelles

Ensuite, on a effectué les mêmes calculs, mais avec Pari en utilisant que les rationnels, c'est à dire, les valeurs de départ sont des données exactes sous formes de rationnels et aucune approximation n'est réalisée au milieu des calculs. Le déterminant est considéré nul au-dessous de la même valeur que précédemment (1E-10).

Tableau 12 : Erreurs absolues maximales pour $out^n(in^n(A))$ en arithmétique rationnelle

| p | $out^1(in^1(A))$ | $out^2(in^2(A))$ | $out^3(in^3(A))$ | $out^4(in^4(A))$ |
|---|---|---|---|---|
| 0,1 | 0 | 0 | 0 | 0 |
| 0,01 | 0 | 0 | 0 | 0 |
| 0,001 | 0 | 0 | 0 | 0 |
| 0,0001 | 0 | 0 | 0 | |
| 0,00001 | 0 | | | |

Tableau 13 : Erreurs absolues maximales pour $in^n(out^n(A))$ en arithmétique rationnelle

| p | $in^1(out^1(A))$ | $in^2(out^2(A))$ | $in^3(out^3(A))$ | $in^4(out^4(A))$ |
|---|---|---|---|---|
| 0,1 | 0 | 0 | 0 | 0 |
| 0,01 | 0 | 0 | 0 | 0 |
| 0,001 | 0 | 0 | 0 | 0 |
| 0,0001 | 0 | 0 | 0 | 0 |
| 0,00001 | 0 | 0 | 0 | 0 |

En utilisant que des valeurs rationnelles, les calculs sont justes, puisqu'ils n'y a pas d'erreur dans les résultats finaux, et ceci quelques soient les valeurs de p ou de n. On remarque aussi que le déterminant est considérer nul à partir de valeurs plus grandes de n et plus petites de p. Ce qui démontre, que dans le cas des calculs effectués en virgule flottante, le déterminant passe au-dessous 1E-10 plus rapidement à cause des erreurs cumulées.

3) Passage valeurs réelles-valeurs rationnelles

Dans un troisième temps, à chaque calcul d'intersection, on est passé des réels aux rationnels pour les calculs et les résultats étaient renvoyés en réels. Le passage réel-rationnel se fait avec une précision relative de l'ordre de 1E-10, sauf pour l'opération $in^n(out^n(A))$ où certains problèmes sont apparus, mais on y reviendra plus en détail après. En ce qui concerne le déterminant, rien ne change.

Tableau 14 : Erreurs absolues maximales pour $in^n(out^n(A))$ avec le passage réel-rationnel-réel

| p | $in^1(out^1(A))$ | $in^2(out^2(A))$ | $in^3(out^3(A))$ | $in^4(out^4(A))$ |
|---|---|---|---|---|
| 0,1 | 0 | 0 | 0 | 2,24E-08 |
| 0,01 | 0 | 8,43E-08 | 4,15E-05 | 2,90E-03 |
| 0,001 | 1E-06 | 1,30E-06 | 3,52E-04 | 3,90E-02 |

Tableau 15 : Erreurs absolues maximales pour $out^n(in^n(A))$ avec le passage réel-rationnel-réel

| p | $out^1(in^1(A))$ | $out^2(in^2(A))$ | $out^3(in^3(A))$ | $out^4(in^4(A))$ |
|---|---|---|---|---|
| 0,1 | 0 | 0 | 8,82E-07 | 1,02E-06 |
| 0,01 | 0 | 2,56E-07 | 1,28E-04 | 2,58E-03 |
| 0,001 | 0 | 2,50E-04 | 5,93E-04 | |
| 0,0001 | 0 | 1,67E-05 | | |
| 0,00001 | | | | |

Les résultats du tableau 15 peuvent être considérés comme peu satisfaisants, mais ils s'expliquent facilement.

En ce qui concerne n = 1, le seul problème se situe pour p = 0,00001, où pour certains calculs, le déterminant a été considéré nul. L'explication est assez simple; quand on effectue l'opération $in^1(A)$, les valeurs de départ ont une erreur maximale de 1E-10 (erreur d'approximation des flottants par des rationnels), les valeurs entières ont bien sûr une erreur nulle. Après le calcul effectué à l'aide des rationnels, on convertit les résultats en flottants, ce qui entraîne aussi une erreur, de l'ordre de la précision du nombre flottant, mais elle existe, et en plus, on a les erreurs d'approximation initiales qui se sont accumulées. Ensuite, on exécute l'opération "going out" sur le résultat calculé, c'est à dire, on calcule $out^1(in^1(A))$. On va alors approcher les coordonnées, calculées juste avant, par des rationnels, mais, comme on avait fait des arrondis, on ne retrouve pas les rationnels calculés lors des intersections. On augmente ainsi les erreurs. D'où le résultat surprenant pour n = 1 et p = 0,00001.

On comprend mieux maintenant, les résultats suivants du tableau. On vient d'expliquer que le passage réel-rationnel-réel accumule volontairement des erreurs, donc, plus on utilise ce passage, plus les erreurs augmentent, ce qui donne de mauvais résultats.

On s'aperçoit aisément que, si on venait à utiliser plusieurs fois de suite des résultats calculés de cette manière, les résultats seraient faussés. Pour remédier à ce problème, il faut conserver le rationnel calculé correspondant au réel cherché, de cette façon, le passage réel-rationnel ne serait effectué qu'une seule fois, on n'aura pas une perte excessive de précision De plus, on gagnerait du temps d'exécution puisque l'on aurait pas à recalculer les rationnels correspondants aux réels.

Le constat est le même pour le tableau 14 et pour les mêmes raisons. Les tests pour p = 1E-04 et 1E-05, n'ont pas pu être effectués au-delà de n = 1, à cause d'un problème d'exécution du programme. Au cours de l'approximation d'un réel par un rationnel, l'algorithme d'Euclide crée des entiers trop grands. D'ailleurs pour p = 1E-02 et p = 1E-03, on a du prendre l'erreur d'approximation égale à 1E-08 et à 1E-06 respectivement. C'est un problème qu'il faudra traiter ultérieurement, soit en utilisant directement Pari, soit en utilisant une autre méthode.

### 5.4. Étude 3

On considère une série de segments aléatoires, auxquels on fait subir des rotations de centre, le milieu des segments, et d'angle variant de 10° à 1E-05°. On calcule pour chaque intersection l'erreur relative entre le résultat calculé avec des rationnels et le résultat théorique (coordonnées de chaque milieu). On effectue alors une étude statistique :

| Angle | Moyenne | Ecart-type |
|---|---|---|
| 10° | 1,885E-09 | 2,273E-09 |
| 5° | 6,096E-09 | 1,504E-08 |
| 1° | 1,439E-08 | 1,660E-08 |
| (1E-01)° | 5,593E-07 | 1,899E-06 |
| (1E-02)° | 1,210E-06 | 1,591E-06 |
| (1E-03)° | 9,731E-06 | 8,486E-06 |
| (1E-04)° | 1,118E-04 | 1,21E-04 |
| (1E-05)° | 5,738E-04 | 6,527E-04 |

On a approché chaque coordonnée des sommets par des rationnels avec une précision de plus ou moins 1E-10. Cette erreur affecte bien entendu le résultat final. Les nouvelles coordonnées calculées lors de la rotation ont des erreurs supplémentaires dues aux calculs en virgule flottante lors de leur détermination. On voit ici que l'erreur initiale se propage au cours des calculs. Cette erreur augmente toujours avec la diminution de l'angle, dans les mêmes proportions que l'étude à l'aide de l'arithmétique d'intervalles. Une fois l'approximation effectuée, il ne demeure aucune erreur due au calcul, la seule opération exécutée en virgule flottante, est le passage du rationnel, correspondant à une coordonnée calculée, en réel. Cette opération tient évidemment compte de l'erreur relative cumulée lors des calculs.

Si l'approximation avait été meilleure, le résultat final l'aurait été aussi. Par exemple, pour une approximation de 1E-14, et avec un angle de (1E-03)°, on obtient une moyenne de l'erreur relative de l'ordre de 2,451E-09.

De plus, lorsque les droites ont subi une rotation d'un angle faible, les valeurs des sommets sont très proches et on a un risque de perte la topologie des objets lors de l'approximation. Si deux sommets flottants sont distincts, on doit produire deux sommets rationnels distincts. Ici, le problème ne sait pas réellement posé puisque le calcul s'est effectué correctement.

Partant de ce constat, il semble possible de pouvoir estimer l'erreur relative du résultat final en tenant compte de l'angle et aussi de l'approximation. On a ici connaissance de l'erreur relative initiale.

## 6. CONCLUSION

Les logiciels de calcul formel permettent d'effectuer des calculs mathématiques exacts de façon symbolique. Ils sont utiles pour la résolution des problèmes formels, car ils évitent les erreurs humaines et sont plus rapides pour ce genre de calcul. Cependant, pour le calcul numérique, ils révèlent être très lents, à cause de la précision qui n'est pas limitée. De plus, la majorité des systèmes de calcul formel ont leur propre langage de programmation et l'interface entre un langage courant, tel que Pascal, C ou C++, est très difficile à mettre en oeuvre.

C'est pourquoi nous nous sommes intéressés à des semi-logiciels de calcul formel, en particulier Pari. Ce système est plus spécialisé dans la théorie des nombres. Il n'offre pas autant de possibilités pour les calculs symboliques, mais il est plus rapide en calcul numérique que les autres logiciels. De plus, l'interface entre Pari et le langage C est facile, en effet, il existe une librairie écrite dans ce langage.

Les logiciels actuels de calculs formels essayent de traiter le plus de problèmes possibles avec une précision très grande, ce qui les rend lents et peu adaptatifs à des modèles de C.A.O. En fait, pour que le calcul formel soit performant, il faut effectuer une étude approfondie de la théorie des nombres et adapter, suivant les besoins, ce qui pourraient améliorer les performances d'un logiciel de C.A.O., de façon à ne pas avoir une contrainte de temps trop importante.

L'utilisation d'une arithmétique exacte permet d'éviter les erreurs de calcul, on le voit très bien avec les deux premiers exemples. L'utilisation de Pari, dans ce contexte, permet d'éviter les problèmes d'espace mémoire que suscite l'utilisation de grands entiers et de ne pas se soucier de l'arithmétique des rationnels. Avec cette arithmétique, on évite les erreurs de calculs (arrondis, perte de chiffres significatifs ...). Quand les données sont exactes, on peut déterminer des points d'intersections sans perte d'informations même pour des angles très petits.

Les erreurs initiales, dues à l'approximation et aux erreurs déjà commises, ne sont pas éliminées, elles se propagent tout au long des calculs. On ne peut pas les éliminer, il faut donc en tenir compte au moment des conclusions.

En C.A.O, il est apprécié que les calculs s'effectuent en temps réel, et il est évident que l'utilisation d'une arithmétique rationnelle ralentit le déroulement des calculs. Donc, pour tirer le meilleur parti d'une telle arithmétique, il faudrait implanter des algorithmes utilisant des nombres flottants qui soient le plus robuste possible et utiliser l'arithmétique rationnelle pour les cas où le risque de mettre l'algorithme en échec est grand.

Il est assez facile d'utiliser ce logiciel à partir de Sacado, cependant, tous les problèmes existants n'ont pas été évoqués ni rencontrés (problèmes d'espace mémoire, de temps ...). De plus, l'arithmétique rationnelle ne résout pas tous les problèmes, elle n'évite pas aux erreurs existantes (celles des approximations en particulier) de se propager.

Nous avons constaté qu'il n'était pas raisonnable de perdre un rationnel calculé en le transformant en réel. Si nous voulons réutiliser le même rationnel, cela est impossible, l'approximation du réel correspondant par les fractions continues ne donnera pas le même résultat. Il convient donc de définir un nouveau type de réel, qui contiendra la valeur en virgule flottante et la valeur du rationnel correspondant s'il a été calculé. Ainsi, nous effectuerons une seule fois l'approximation (quand nous estimerons que c'est nécessaire) du réel par un rationnel, ce qui nous fera gagné du temps. Et si l'emploi du rationnel n'est pas nécessaire, nous utiliserons la valeur en virgule flottante.

# CHAPITRE V

# LA MÉTHODE E.C.P

# EN C.A.O

# 1. INTRODUCTION

Comme nous l'avons vu aux deux premiers chapitres, les erreurs numériques dues à l'utilisation de l'arithmétique flottante provoquent des erreurs topologiques ou même des erreurs d'exécution. Afin d'améliorer la maîtrise des ces erreurs et de les diminuer, nous proposons une méthode que nous appellerons désormais la méthode "**E.C.P.**" en C.A.O. (**E**stimation et **C**ontrôle de la **P**récision).

Pour utiliser cette méthode, nous redéfinissons tout d'abord le réel informatique (représenté par les types float ou double dans le langage C). La nouvelle représentation nous permet d'avoir, pour chaque réel, sa valeur en nombre flottant, le nombre de chiffres significatifs qui lui est associé et le nombre rationnel qui lui est rattaché si celui-ci a été déterminé.

Nous rappelons ensuite la méthodologie qui nous permet de calculer le nombre de chiffres significatifs d'un réel obtenu au cours d'un calcul, et si ce nombre nous paraît trop faible, nous recourons à l'arithmétique rationnelle.

Nous proposons également de modifier le concept de décidabilité pour les instructions conditionnelles utilisées couramment, de manière que ces dernières tiennent compte des erreurs cumulées au cours des calculs.

# 2. LA MÉTHODE E.C.P.

## 2.1. Introduction

Au cours des calculs effectués, l'algorithme doit déterminer la précision des résultats. Nous avons vu, au chapitre III, qu'il existait une méthode stochastique qui permet d'estimer les erreurs d'arrondi au cours des calculs tout en tenant compte des erreurs sur les données. Nous allons donc utiliser ce procédé pour déterminer la précision des résultats dans notre méthode (**E**stimation et **C**ontrôle de la **P**récision) qui se constitue principalement de trois phases pour la détermination de la précision :

- Initialisation des données
- Calcul du nombre de chiffres significatifs lors des calculs
- Utilisation de l'arithmétique rationnelle

## 2.2. Initialisation des données

Lors de la saisie ou de l'inititialisation de valeurs (coordonnées de points, vecteurs ...), les données peuvent être supposées exactes, puisqu'elles ne proviennent d'aucun calcul.

Il suffit donc de faire correspondre au nombre de chiffres significatifs du réel, la tolérance correspondant à une donnée exacte, par exemple, la valeur maximale de chiffres significatifs d'un nombre flottant.

Nous devons aussi savoir à n'importe quel moment du programme, si le rationnel correspondant au nombre flottant a été calculé ou non. Pour ce faire, nous utilisons la convention suivante :

- Si le rationnel n'a pas été estimé, le premier élément du tableau correspondant au dénominateur (den) contiendra la valeur zéro, sinon, il contiendra la longueur effective de l'entier.

### 2.3. Calcul du nombre de chiffres significatifs lors des calculs

Durant l'exécution d'un programme, de nouvelles données sont créées. Elles proviennent souvent de transformations ou de calculs d'intersection de différents objets. Ces créations étant effectuées par un calcul en arithmétique flottante, l'apparition d'erreur ou de risque d'annulation de digits (cf Chapitre I) est omniprésente. Il convient donc d'utiliser la méthode CESTAC pour déterminer le nombre de chiffres significatifs de ce nouveau réel. Nous ne redétaillons pas le principe de la méthode, car il a été détaillé au chapitre III, nous énonçons seulement les grandes lignes.

Nous nous plaçons dans le cas d'un simple calcul, c'est à dire, qu'il n'est pas nécessaire d'effectuer un choix (instructions conditionnelles). Dans cette situation, l'implémentation asynchrone (cf Chapitre III § 3.5) de la méthode est suffisante.
Nous effectuons donc trois fois le calcul en perturbant les données et en tenant compte des erreurs affectant les données initiales. Nous obtenons ainsi, avec l'aide du module NCSE, le nombre de chiffres significatifs du résultat. Il suffit alors d'affecter à la variable résultat de type réel, la valeur trouvée, le nombre de chiffres significatifs et de mettre à zéro le premier élément du tableau den (dénominateur).

### 2.4. Utilisation de l'arithmétique rationnelle

Dans le paragraphe précédent, nous conservons le résultat et le nombre de chiffres significatifs qui lui correspondent, sans se soucier s'il est correct ou non. Il est nécessaire d'essayer d'éviter ces désagréments, c'est pourquoi nous proposons d'avoir recours à l'arithmétique rationnelle lorsque cela s'avère nécessaire. Le seuil sous lequel nous considérons que le résultat ne nous convient pas dépend de la précision que nous voulons conserver. Il ne nous semble pas très satisfaisant de faire intervenir l'arithmétique rationnelle que dans le cas où le nombre de chiffres significatifs est nul. Il est préférable d'essayer de conserver un minimum de chiffres significatifs (mincs).

Pour ce faire, il suffit de tester le nombre de chiffres significatifs du résultat, s'il est correct, alors le résultat est conservé simplement comme dans le paragraphe précédent, sinon, nous réeffectuons les calculs avec les nombres rationnels.

# 3. MISE EN OEUVRE

## 3.1. Introduction

L'idée principale de notre méthode étant de pouvoir maîtriser les erreurs, ainsi que d'améliorer les tests utilisant des tolérances, l'algorithme, au cours de son exécution doit donc savoir quelle est la précision des réels.

Pour ce faire, nous allons associer à chaque réel utilisé, le nombre de chiffres significatifs avec lequel il est représenté. Ainsi, lors de la comparaison de deux nombres, les tests pourront être effectués par rapport aux nombres de chiffres significatifs de chaque réel, et non plus par rapport à une tolérance, souvent aléatoire.

Nous voulons aussi avoir recours à l'arithmétique exacte, pour cela, nous suggérons de conserver le numérateur et le dénominateur du rationnel approchant le réel lorsqu'il aura été déterminé par la méthode d'approximation choisie ou lorsqu'il est le résultat d'un calcul en arithmétique rationnelle. De cette façon, nous ne calculons qu'une seul fois au maximum cette approximation. En effet, si un réel a été, pour des raisons de précision, approché par un rationnel, et s'il s'avère nécessaire de réutiliser cette approximation dans l'exécution de l'algorithme, nous évitons ainsi de recommencer deux fois le même calcul. De plus, si le rationnel provient d'un calcul en arithmétique exacte, et si nous ne le conservons pas, l'approximation du réel correspondant ne fournit pas forcément le rationnel initial.

## 3.2. Nouvelle définition du réel

### *3.2.1. Formulation du problème*

Nous rappelons que l'arithmétique rationnelle est réalisée grâce au semi-logiciel de calcul formel Pari (cf. Chapitre IV). Ce logiciel permet d'éviter le problème de la taille des entiers et la création individuelle d'une arithmétique rationnelle avec des entiers de tailles quelconques.

Les rationnels du logiciel Pari (que nous appellerons des rationnels de type GEN dans la suite, type fondamental utilisé pour tous les objets de Pari, il en sera de même pour les entiers), sont composés de deux parties : le numérateur et le dénominateur. Ces deux entités sont des entiers de type GEN.

Pour conserver le rationnel dans le nouveau type réel, nous allons donc conserver le numérateur, le dénominateur, ainsi que le signe. La conversion d'un entier de type GEN en un entier de type long (en C) peut se faire facilement à l'aide des fonctions de Pari, dans la mesure où l'entier GEN ne dépasse pas la taille acceptée par le type long (soit 32 bits). Mais ceci n'est pas forcément vérifié et si nous ne faisons pas attention, l'exécution du programme échouera. Afin de remédier à ce problème, nous allons stocker les entiers GEN dans des tableaux d'entiers long. Ceci est facilement réalisable grâce à la structure de l'entier GEN.

*3.2.2. Structure du réel*

La nouvelle structure du réel que nous définissons est donc la suivante :

```
réel =   Enregistrement
           x    : double;                 { valeur du réel }
           cs   : double;                 { nombre de chiffres significatifs }
           num  : tableau d'entiers long  { numérateur du rationnel approchant x }
           den  : tableau d'entiers long  { dénominateur du rationnel approchant x }
           s    : entier long             { signe du rationnel }
         FinEnregistrement
```

Il nous faut maintenant créer trois algorithmes pour gérer ce réel : le premier permet de conserver le rationnel GEN dans les tableaux correspondant au numérateur et au dénominateur dans le type réel, le deuxième et le troisième correspondent à la manoeuvre inverse. Ils créent deux entiers GEN correspondant au numérateur et au dénominateur, et permettent la création du rationnel GEN.
La description des ces trois algorithmes est détaillée en annexe 12.

**3.3. Nouveau concept de décidabilité**

*3.3.1. Introduction*

Dans la plus part des cas de tests, on utilise une tolérance à partir de laquelle on peut considérer que deux nombres sont égaux ou qu'une valeur est nulle. C'est souvent au cours des ces instructions conditionnelles que des erreurs interviennent. Afin de résoudre ce problème, nous allons apporter une nouvelle définition pour les comparaisons réalisées dans les logiciels de C.A.O. Elle s'intègre directement dans la méthode E.C.P que nous développons.
Deux cas se présentent :

- Le test s'effectue entre deux données, dont au moins une provient d'un calcul. Dans ce cas-là, nous reprenons l'idée qui a été développée aux paragraphes 3.6. et 3.7. du chapitre III, c'est-à-dire, que nous utilisons une implémentation synchrone de la méthode CESTAC de manière à connaître la précision du nombre à comparer.
- Le test s'effectue entre deux données ne provenant pas directement de calculs. Nous employons les précisions associées aux nombres flottants pour réaliser la comparaison.

*3.3.2. Les instructions conditionnelles*

Au cours d'un calcul, l'algorithme peut, à un certain moment, être conduit à prendre une décision, et suivant la réponse du test, il poursuit une séquence de calculs ou une autre. C'est souvent durant de tels tests que des erreurs interviennent. Pour minimiser ce type d'erreurs,

nous utilisons l'implémentation synchrone de la méthode CESTAC. Plus précisément, nous voulons tenir compte des erreurs affectant les valeurs intervenant dans le test.

Nous avons déjà présenté au paragraphe 3.6. du chapitre III les différents cas pouvant intervenir lors de la comparaison de deux nombres. Nous développons ici les concepts qui y ont été exposés.

Les deux principaux tests intervenants dans les algorithmes sont le test de nullité d'un nombre et la comparaison entre deux nombres.

***a) Test de nullité***

Plaçons-nous dans le cas où l'on veut savoir si un résultat R d'un calcul est nul ou non. Le test mathématique est bien évidemment le suivant :
si R = 0 alors instruction 1 sinon instruction 2

Nous savons qu'il est informatiquement impossible d'effectuer ce test, car l'obtention du zéro exact est, dans la plus par des cas, irréalisable. Le test informatique utilisé couramment est :
si R < eps alors instruction 1 sinon instruction 2

Si l'obtention du résultat R introduit une erreur d'arrondi supérieure à la tolérance eps, alors l'instruction 2 sera exécutée même si cela est incorrect. Pour tenir compte de l'éventuelle erreur d'arrondi, nous utilisons l'implémentation synchrone de la méthode CESTAC.

A l'aide de cette méthode, à la fin des calculs précédant le test, nous avons la moyenne RM du résultat R et le nombre de chiffres significatifs CRM qui lui est associé, ainsi que la précision des deux opérandes intervenant au dernier calcul pour l'obtention de R, com1 et com2 (nombre de chiffres significatifs des opérandes om1 et om2) .Trois cas se présentent :

- Les valeurs om1 ou om2 et RM sont significatives, en programmation asynchrone, ce serait toujours la même séquence qui serait exécutée. Il n'y a donc aucune ambiguïté, le programme peut continuer sans problème.

- Les valeurs om1 et om2 sont significatives, mais pas la valeur RM. C'est le cas du zéro informatique, c'est-à-dire, que la valeur de RM ne représente que l'effet cumulé des erreurs. Dans ce cas, nous considérons que RM est nul et le programme poursuit l'instruction 1.

- Les valeurs om1 et om2 sont non significatives, nous sommes confrontés au cas d'indécidabilité. Nous préconisons alors de réeffectuer les calculs avec l'arithmétique rationnelle. (Algorithme annexe 13)

***b) Tests de comparaison***

Le procédé est quasi identique au précédent. En effet, comparer deux valeurs A et B revient à comparer A – B avec zéro. Nous ne développons pas les algorithmes utilisés, mais nous présentons les manipulations à effectuer.

Dans la suite du paragraphe, les variables AM, BM, CAM, CBM, DM et CDM correspondent respectivement aux moyennes représentant les valeurs A et B et leur nombre de chiffres significatifs ainsi la moyenne de la différence entre A et B et le nombre de chiffres significatifs correspondant.

Nous désirons effectuer la comparaison A op B. Ceci équivaut à effectuer la comparaison A – B op 0, où op est un opérateur de comparaison ($<$ , $\leq$ , $>$ , $\geq$).

Nous sommes confrontés à trois cas comme précédemment :

- Les valeurs CAM ou CBM et CDM sont significatives, nous comparons CDM et zéro avec l'opérateur op, et le programme se poursuit normalement.

- Les valeurs CAM et CBM sont significatives, mais pas CDM. Nous considérons que A = B.

- Les valeurs CAM et CBM sont non significatives, nous faisons appel à l'arithmétique exacte pour une détermination plus précise de A et B.

*3.3.3. Caractérisation des opérateurs de comparaison*

L'égalité entre réels ne peut se faire, sauf cas particulier, à l'aide de l'égalité existant dans les langages. Ainsi, l'utilisation des fonctions booléennes pour exprimer les relations d'égalité et inégalité entre réels est courante. Les tests étaient effectués par rapport une tolérance qui était déterminée en fonction de la précision de la machine, mais qui ne tenait pas compte des erreurs d'arrondi.

Grâce à la nouvelle définition du réel, nous avons en permanence la précision de chaque nombre flottant. Nous définissons une nouvelle conception de comparaison qui consiste à :

- perturber trois fois les deux données à comparer
- effectuer leur différence et nous obtenons un tableau de trois valeurs
- nous déterminons la moyenne de ces trois valeurs et le nombre de chiffres significatifs correspondant
- il suffit alors de regarder si la différence est significative, dans ce cas nous regardons le signe de la différence, sinon nous supposons que les deux valeurs sont égales.

A l'aide de ces trois fonctions, détaillée en annexe 13, nous pouvons estimer une comparaison en tenant compte des erreurs d'arrondi accumulées.

Nous venons de présenter notre méthodologie qui permet une meilleure maîtrise des erreurs numériques et surtout celles dues aux arrondi qu'effectue l'ordinateur. Nous présentons maintenant quelques exemples d'exécution de la méthode E.C.P.

## 4. EXEMPLES

Nous proposons de vérifier que la méthode, que nous proposons, permet de connaître la précision d'un calcul. Pour cela, nous reprenons l'exemple du pentagone décrit au paragraphe 2. du chapitre I. Cet algorithme n'est évidemment basé que sur des intersections de droites. L'algorithme utilisé est implémenté de manière synchrone, à cause du test sur le déterminant. Ainsi, il sera considéré nul s'il ne représente que l'effet cumulé des erreurs (plus de tolérance).

**Algorithme d'intersection avec une implémentation synchrone**

entrée : les sommets $x_i$, $y_i$ de type "réel"

sortie : le point d'intersection de type "réel", s'il a pu être calculé

```
début
  pour k de 1 à 3 faire
    perturbation des données x_i, y_i
    détermination et perturbation des coefficients a_i, b_i, c_i des deux droites
    calcul de m_1 et m_2 et du déterminant (det = m_1 – m_2)
  fpour
  calcul du nombre de chiffres significatifs cm_1 et cm_2 des moyennes m_1m et m_2m
  de m_1 et m_2
  si ( cm1 < 1 et cm2 < 1 ) alors utilisation d'une arithmétique rationnelle
  sinon
    calcul du nombre de chiffres significatifs cdet de la moyenne cdetm de det
    si ( cdetm < 1 ou detm = 0 ) alors pas d'intersection (zéro informatique)
    sinon
      pour k de 1 à 3 faire
        calcul de x et y avec l'arithmétique perturbée
      fpour
      calcul du nombre de chiffres significatifs cxm et cym de la moyenne cxm et cym
      si ( cxm <  mincs ou cym < mincs ) alors utilisation d'une arithmétique rationnelle
      sinon xm et ym sont solutions et cxm et cym sont leur nombres de chiffres significatifs
      fsi
    fsi
  fsi
fin
```

Tableau 15 : Opération $out^n in^n( A))$

| p | $out^1in^1( A))$ | | | $out^2(in^2( A))$ | | | $out^3(in^3( A))$ | | | $out^4(in^4( A))$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ER | E | CS | ER | E | CS | ER | E | CS | ER | E | CS |
| 0,1 | 1E-15 | 1E-16 | 14,3 | 6E-15 | 4E-15 | 12,6 | 2E-13 | 7E-12 | 10,2 | 4E-13 | 7E-10 | 8 |
| 0,01 | 0 | 1E-16 | 14,2 | 6E-12 | 3E-14 | 13.0 | 9E-11 | 4E-08 | 6,1 | 5E-08 | 3E-04 | 4,4 |
| 0,001 | 1E-13 | 1E-16 | 14,2 | 4E-10 | 1E-11 | 11,8 | 4E-07 | 8E-07 | 7,7 | | | |
| 0,0001 | 1E-13 | 1E-16 | 14,3 | 2E-10 | 3E-09 | 10,9 | | | | | | |

Tableau 16 : Opération $in^n out^n( A))$

| p | $in^1(out^1( A))$ | | | $in^2(out^2( A))$ | | | $in^3(out^3( A))$ | | | $in^4(out^4( A))$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ER | E | CS | ER | E | CS | ER | E | CS | ER | E | CS |
| 0,1 | 0 | 1E-16 | 14,3 | 2E-15 | 5E-15 | 12,6 | 2E-13 | 8E-12 | 10,2 | 6E-12 | 7E-10 | 8 |
| 0,01 | 0 | 1E-16 | 14,2 | 1E-14 | 3E-14 | 13.0 | 5E-11 | 5E-12 | 10 | 8E-08 | 3E-04 | 5,6 |
| 0,001 | 0 | 1E-16 | 15,4 | 1E-10 | 6E-14 | 12,1 | 9E-08 | 3E-07 | 8 | 2E-01 | 4E-06 | 7,4 |
| 0,0001 | 0 | 1E-16 | 15,0 | 1E-08 | 8E-13 | 10,9 | 7E-05 | 1E-16 | 14,4 | 8E00 | 5E-04 | 7,2 |

Nous présentons comme résultats, les plus grandes erreurs commises (ER) pour chaque pas commises par l'arithmétiques flottantes, les plus grandes erreurs commises (E) pour chaque pas avec la méthode E.C.P. et l'estimation de la précision des résultats (CS) correspondant à cette erreur. Plusieurs constats découlent de ces résultats et ils donnent lieu à quelques explications. Nous préférons commenter les deux tableaux à part pour plus de clarté.

Le tableau 15 représente les résultats pour le cas où l'opération in( ) est effectuée en premier. En comparant la précisions des résultats par rapport aux résultats déterminés par une arithmétique flottante, nous voyons que pour n = 1, nous ne perdons pas de précision, l'algorithme ayant fait appel, à certains moments, à l'arithmétique exacte. Pour n = 2, le constat est identique, nous gagnons en précision. De plus, nous remarquons que l'on obtient le nombre de chiffres significatifs avec une précision assez correcte. Pour n = 3 et n = 4, la précision n'est pas meilleure, mais nous avons à notre disposition la précision des résultats. Ceci peut s'expliquer facilement; à chaque étape, nous perturbons les données en fonction de leur précision, et plus la profondeur est grande, plus grande est la précision perdue et les calculs sont effectués avec des données dont les perturbation sont plus élevées.
En comparant maintenant, par rapport aux résultats du tableau 15 (Chap IV §5.3.), correspondant au cas où est effectué continuellement le passage réel-rationnel-réel, les résultats sont nettement meilleurs. Ceci montre bien l'intérêt de conserver les rationnels déterminés par des calculs.

Pour le tableau 16, le constat est simple, les résultats sont nettement meilleurs, que ce soit pour l'arithmétique flottante ou pour le passage réel-rationnel-réel (Tableau 14 Chap IV §5.3.).

Nous reprenons aussi le test effectué dans l'étude 1 du paragraphe 4.3 du chapitre II. Nous considèrons le segment d'extrémités [-20, -20] et 30, 30] auquel nous effectuons une suite de rotations avec un angle de plus en plus petit, et nous déterminons les intersections entre le segment initial et ceux construits par rotations.

| ANGLE | INTERSECTION | EX | CS |
|---|---|---|---|
| (1E-01)° | oui | 1,3E-12 | 11,8 |
| (1E-02)° | oui | 8.4E-11 | 11,3 |
| (1E-03)° | oui | 0 | 16 |
| (1E-04)° | oui | 0 | 16 |
| (1E-05)° | oui | 0 | 16 |
| (1E-06)° | oui | 0 | 16 |
| (1E-07)° | oui | 0 | 16 |

La valeur EX représente l'erreur su l'abscisse (celle de l'ordonnée étant identique) et CS et le nombre de chiffres significatifs correspondant. Pour les angles de (1E-01)° et (1E-02)°, l'algorithme ne fait pas appel à l'arithmétique rationnelle, car la précison est supposée suffisante. Pour les autres angles, les chiffres significatifs des termes intervenants dans le calcul du déterminant ne sont plus significatifs ou ceux des coordonnées calculées, il y a alors appel de l'arithmétique exacte. Dans ce cas, on retrouve évidemment les valeurs obtenues au chapitre IV, c'est à dire, il n'y a plus d'erreur.

## 5. CONCLUSION

Nous venons de présenter une méthode permettant de déterminer le nombre de chiffres significatifs du résultat d'une suite de calcul. Elle permet également d'effectuer ces calculs à l'aide de l'arithmétique rationnelle lorsque l'arithmétique flottante a introduit des erreurs d'arrondi qui mettent à défaut le bon déroulement de l'algorithme. Nous avons été obligés de redéfinir le notion de réel, afin de connaître la précision de chaque nombre flottant, ainsi que le rationnel qui lui est associé.

Connaître en permanence la précision de nombres flottants, nous permet de modifier les tests de décidabilité et de comparaison. Ces derniers ne sont plus effectués par rapport à une précision quelconque, mais par rapport à leur précision. Ainsi, nous considérons comme nulle, une valeur qui ne représente en fait que les erreurs cumulées.

Cette méthode permet donc de maîtriser, et même diminuer (en particulier dans où les erreurs d'arrondi sont trop importantes) les erreurs numériques et d'estimer la précision d'un résultat. Il suffit pour constater cela, de comparer les résultats obtenues par cette méthode avec les résultats des chapitres précédents.

# CONCLUSION

## CONCLUSION

Cette thèse est consacrée aux problèmes d'imprécision numérique dus à l'arithmétique en virgule flottante. L'objectif de ce travail a été d'introduire une solution permettant de connaître la précision d'un calcul. Il se décompose en cinq parties.

Dans un premier temps, il nous a semblé nécessaire d'analyser les diverses solutions proposées dans la littérature. Cet état de l'art constitue le chapitre I. Nous rappelons tout d'abord les notions de base de l'arithmétique en virgule flottante, ainsi que les différents types d'erreurs liés à cette arithmétique et nous présentons quelques exemples de défauts intervenant au cours d'algorithmes de C.A.O.

Ensuite, nous résumons chaque méthodologie en exposant les avantages et inconvénients de chacune. Il apparaît que, quelque soit la méthode suggérée, l'amélioration de précision nécessite inévitablement une augmentation de temps et parfois d'espace mémoire. Certaines vont même jusqu'à perdre les données topologiques (§2.), cette méthode est bien sur à éviter. La plus motivante de ces méthodes est celle qui lie l'arithmétique rationnelle à l'arithmétique flottante présentée sous le nom "d'arithmétique paresseuse" [BEN 93].

Une fois cette analyse bibliographique réalisée, nous avons étudié, dans le chapitre II, la propagation des erreurs au cours d'algorithmes géométriques. Nous avons choisi d'examiner le cas particulier du calcul d'intersection de droites, car c'est un algorithme utilisé couramment en C.A.O. Nous montrons, à l'aide d'un exemple, que l'analyse différentielle n'est pas suffisante pour déterminer l'erreur qui entache un résultat. Elle permet seulement de connaître l'influence des erreurs de données sur le résultat, mais elle ne prend pas en compte les erreurs d'arrondi que commet l'ordinateur.

Pour tenir compte de ces erreurs, nous avons réalisé une étude à l'aide de l'arithmétique d'intervalle. Cette étude a permis d'obtenir une estimation de la perte de chiffres significatifs au cours de la détermination de point d'intersection de deux droites, en fonction de l'angle qu'elles forment.

Le troisième chapitre présente la méthode CESTAC qui permet de déterminer le nombre de chiffres significatifs du résultat d'un calcul. L'implémentation synchrone de cette méthode permet de gérer correctement les instructions conditionnelles d'une procédure informatique et donc de contrôler le bon déroulement du programme de calcul.

Comme l'arithmétique flottante n'est qu'une approximation de l'arithmétique exacte, il est utile d'introduire un nouveau concept, celui du zéro informatique. Ce concept transcende la notion de valeur nulle pour atteindre celle de valeur non significative, c'est-à-dire de valeur entachée d'une erreur d'arrondi du même ordre de grandeur que la valeur numérique elle-

même. Le zéro informatique permet de contrôler la validité de la solution fournie par la machine.

Le quatrième chapitre est consacré à l'arithmétique rationnelle, qui permet d'éviter les erreurs d'arrondi. Travailler avec des rationnels nécessite un développement algorithmique important, auquel il faut consacrer une étude complète, afin d'optimiser et de rendre performante cette arithmétique. Ce problème n'étant pas une priorité pour ce travail, nous avons voulu utilisé des logiciels existants tels que les logiciels de calcul formel. Le problème d'une interface entre un tel logiciel et le modèle SACADO, nous a amené à employer le logiciel de calcul formel PARI.

La méthode que nous proposons, dans le chapitre V, permet d'associer les deux principes développés précédemment. Nous voulons avoir recours à l'arithmétique rationnelle lorsque le résultat d'un calcul a un nombre de chiffres significatifs (déterminés par la méthode CESTAC) supposé insuffisant. Pour nous permettre de réaliser ce nouveau concept, nous redéfinissons la notion de réel. Le nouveau type que nous définissons permet d'associer au nombre en virgule flottante le nombre de chiffres significatifs qui lui est associé. De plus, si le rationnel qui approche ce nombre a été déterminé (soit par la méthode des fractions continues, soit au cours d'un calcul), la nouvelle représentation permet de conserver et d'avoir à notre disposition ce rationnel.

A partir de cette nouvelle définition, nous avons redéfini les notions de décidabilité, à l'aide du zéro informatique, et les notions de comparaisons, qui désormais, tiennent compte de la précision des valeurs comparées.

Beaucoup de points important n'ont pas été traités, en particulier celui du temps. Il est important, en C.A.O., que les calculs s'effectuent en temps réel. Ce point capital n'a pas été exclu des pensées, mais il nous est apparu important de proposer dans un premier temps une méthode, et de l'améliorer correctement dans la suite. Deux points essentiels d'optimisation apparaissent, le premier concerne l'arithmétique rationnelle. L'utilisation d'un logiciel intermédiaire ralentit le processus de calcul et il est intéressant de développer sa propre arithmétique rationnelle et c'est un sujet qui est d'actualité au Laboratoire. Le deuxième point concerne la méthode CESTAC qui, pour déterminer le nombre de chiffres significatifs, effectue trois fois les mêmes calculs avec des données différentes. Une programmation en parallèle du type SIMD (Simple Instruction Multiple Data), peut permettre d'améliorer les performances de cette méthode, car elle permet d'exécuter, sur des processeurs différents, les mêmes instructions avec des données différentes, ce qui est le cas pour la méthode des perturbations.

Il reste aussi maintenant à appliquer cette méthode à d'autre algorithme de C.A.O., et pourquoi pas, de pouvoir l'intégrer ensuite au nouveau modèle REGAIN [GAR 94], développé au Laboratoire.

# ANNEXES

## ANNEXE 1 : Algorithme de Bentley-Ottmann

L'algorithme de Bentley-Ottmann calcule les intersections des arêtes, à chaque nouveau point d'intersection détecté, un nouveau sommet est créé et les deux arêtes de l'intersection sont coupées en deux. Il utilise deux ordres :

1. l'xy-ordre qui classe les sommets par abscisses croissantes; à abscisses égales, les sommets sont classés par ordonnées croissantes. On différencie alors les deux sommets d'une arête, le plus petit se trouve "à gauche" et le plus grand "à droite".
2. l'y-ordre classe les arêtes ( dites "actives" ) sur l'ordonnée croissante de leur point d'intersection avec la droite de balayage ( droite parallèle à l'axe des y ).

L'algorithme est le suivant :

```
Bentley-Ottmann;
début    { Bentley-Ottmann }
   insérer dans xy-ordre tous les sommets  { classement des sommets selon xy-ordre }
   initialiser y-ordre à vide              { droite de balayage est vide au départ,
                                             puis elle balaie tous les sommets }
   pour chaque sommet S, selon xy-ordre, faire
      si S est :
         - un point de mort :              { toutes les arêtes incidentes en S meurent en S }
            enlever les arêtes mortes de y-ordre;
            comparer(active-basse, active-haute)
{ ce sont respectivement les arêtes immédiatement au-dessous et au-dessus du sommet S}
         - un point de naissance :         { toutes les arêtes incidentes en S naissent en S }
            insérer les arêtes naissantes dans y-ordre;
            comparer(la plus basse des nouvelles-nées, actives-basse)
            comparer(la plus haute des nouvelles-nées, actives-haute);
         - un point de vie :               { autres types de sommets }
            enlever les arêtes mortes de l'y-ordre;
            les remplacer par les arêtes naissantes;
            comparer(la plus basse des nouvelles-nées, actives-basse)
            comparer(la plus haute des nouvelles-nées, actives-haute);
      fsi
   fpour
fin
```

```
comparer ( a, b : arêtes );
début    { comparer }
   si  (a et b existent) et
   (a et b convergent vers la droite) et
   (leurs sommets droits sont distincts) et
   ( et b se recouvrent en y) alors
      i ← point d'intersection de a et b;
      si (i existe) alors
         mettre à jour les structures de données de CP;
         insérer i dans xy-ordre;
      fsi
   fsi
fin
```

## ANNEXE 2 : Algorithme d'intersection de droites

### Intersection de deux segments

entrée : coordonnées des sommets des deux segments
sortie : le point d'intersection s'il existe, un entier int précisant le type d' intersection
début *{ interseg }*
si les boîtes englobantes ne se coupent pas alors pas d'intersection, int ← 0
sinon *{ calcul des coefficients des deux droites $a_i$, $b_i$, $c_i$ et du déterminant det}*
si (abs(det) < eps) alors pas d'intersection, int ← 0
sinon les droites se coupent
$x \leftarrow (c_2.b_1 - c_1.b_2)/det$, $y \leftarrow (c_1.a_2 - c_2.a_1)/det$, int ← 1
fsi
si le point appartient aux deux segments alors int ← 1 sinon int ← 0
fsi
si on n'a pas déterminé d'intersection
alors on regarde si les segments se superposent
fsi
fsi
fin

### Superposition des segments

entrée : coordonnées des sommets des deux segments
sortie : entier int précisant la partie superposée s'il y en a une
début
t1 ← appar_seg($x_3,y_3,x_4,y_4,x_1,y_1$) *{ on teste si le point défini par les deux }*
t2 ← appar_seg($x_3,y_3,x_4,y_4,x_2,y_2$) *{ premières coordonnées appartient au }*
t3 ← appar_seg($x_1,y_1,x_2,y_2,x_3,y_3$) *{segment défini par les deux autres points }*
t4 ← appar_seg($x_1,y_1,x_2,y_2,x_4,y_4$) *{ou si il est égal à l'un des deux sommets }*
si ( t1=0) et (t2=0) et (t3=0) et (t4=0) alors pas d'intersection, int ← 0
sinon
si (t1*t2 > 0) alors int ← 2 sinon *{ partie superposée $(x_1,y_1),(x_2,y_2)$ }*
si (t1*t3 > 0) alors int ← 3 sinon *{ partie superposée $(x_1,y_1),(x_3,y_3)$ }*
si (t1*t4 > 0) alors int ← 4 sinon *{ partie superposée $(x_1,y_1),(x_4,y_4)$ }*
si (t2*t3 > 0) alors int ← 5 sinon *{ partie superposée $(x_2,y_2),(x_3,y_3)$ }*
si (t2*t4 > 0) alors int ← 6 sinon *{ partie superposée $(x_2,y_2),(x_4,y_4)$ }*
si (t2*t4 > 0) alors int ← 7 *{ partie superposée $(x_3,y_3),(x_4,y_4)$ }*
fsi
fin

**Appartenance d'un point à un segment :**

entrée : coordonnées des sommets du segment et du point

sortie : la fonction de type entier renvoyant la nature de l'appartenance

début *{ appar_seg }*

$d_1 \leftarrow \sqrt{(x-x_1)^2+(y-y_1)^2}$

$d_2 \leftarrow \sqrt{(x-x_2)^2+(y-y_2)^2}$

$d_3 \leftarrow \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

si ( abs($d_1$+$d_2$-$d_3$) < eps ) alors le point appartient au segment, appar_seg ← 1
sinon le point n'est pas à l'intérieur du segment, appar_seg ← 0

si ( $d_1$ < eps )
alors le point est confondu avec l'origine du segment, appar_seg ← 2

si ( $d_2$ < eps )
alors le point est confondu avec l'extrémité du segment, appar_seg ← 3

fin

## ANNEXE 3 : Calcul de l'abscisse x

### Calcul du numérateur $n_1$ :

- On encadre dans un premier temps les variables intervenant dans le calcul de x :

$$a_1 - \varepsilon(|y_1|+|y_2|) \le a_1 \le a_1 + \varepsilon(|y_1|+|y_2|)$$
$$a_2 - \varepsilon(|y_3|+|y_4|) \le a_2 \le a_2 + \varepsilon(|y_3|+|y_4|)$$
$$b_1 - \varepsilon(|x_1|+|x_2|) \le b_1 \le b_1 + \varepsilon(|x_1|+|x_2|)$$
$$b_2 - \varepsilon(|x_3|+|x_4|) \le b_2 \le b_2 + \varepsilon(|x_3|+|x_4|)$$

- $c_1 - \varepsilon(M_1 + M_2) \le c_1 \le c_1 + \varepsilon(M_1 + M_2)$

avec

$$M_1 = \max\left(-a_1|x_1| + x_1(|y_1|+|y_2|), -a_1|x_1| - x_1(|y_1|+|y_2|), a_1|x_1| + x_1(|y_1|+|y_2|), a_1|x_1| - x_1(|y_1|+|y_2|)\right)$$

$$M_2 = \max\left(-b_1|y_1| + y_1(|x_1|+|x_2|), -b_1|y_1| - y_1(|x_1|+|x_2|), b_1|y_1| + y_1(|x_1|+|x_2|), b_1|y_1| - y_1(|x_1|+|x_2|)\right)$$

Comme on recherche le maximum de quatres valeurs de la forme :
-A - B, -A + B, A - B, et A + B, ce sera forcément $|A|+|B|$

donc

$$M_1 = \left|a_1.|x_1|\right| + \left|x_1(|y_1.|+|y_2|)\right| \le |a_1|.|x_1| + |x_1|.\left|(|y_1.|+|y_2|)\right|$$
$$\le (|y_1.|+|y_2|).|x_1| + |x_1|.\left|(|y_1.|+|y_2|)\right| \le 2.|x_1|.\left|(|y_1.|+|y_2|)\right|$$

On pose :

$$\boxed{M_1 = 2|x_1|(|y_1|+|y_2|)}$$

De la même manière,

$$\boxed{M_2 = 2|y_1|(|x_1|+|x_2|)}$$

- $c_2 - \varepsilon(M_3 + M_4) \le c_2 \le c_2 + \varepsilon(M_3 + M_4)$

où

$$M_3 = \max\left(-a_2|x_3| + x_3(|y_3|+|y_4|), -a_2|x_3| - x_3(|y_3|+|y_4|), a_2|x_3| + x_3(|y_3|+|y_4|), a_2|x_3| - x_3(|y_3|+|y_4|)\right)$$

$$M_4 = \max\left(-b_2|y_3| + y_3(|x_3|+|x_4|), -b_2|y_3| - y_3(|x_3|+|x_4|), b_2|y_3| + y_3(|x_3|+|x_4|), b_2|y_3| - y_3(|x_3|+|x_4|)\right)$$

Pour les mêmes raisons évoquées lors du calcul de $M_1$ et $M_2$, on obtient :

$$\boxed{M_3 = 2|x_3|(|y_3|+|y_4|)}$$
$$\boxed{M_4 = 2|y_3|(|x_3|+|x_4|)}$$

Remarque : Dans la suite des calculs, ainsi que dans les annexes suivantes, il ne sera pas rappelé la simplifaction de max (-A - B, -A + B, A - B, et A + B), elle sera supposée évidente.
Quand la simplification ne sera pas évidente, elle sera détaillée.

•On a donc après calculs que :

$$\boxed{n_1 - \varepsilon(M_5 + M_6) \le n_1 \le n_1 + \varepsilon(M_5 + M_6)}$$

où

$$M_5 = \max\big(-c_2(|x_1|+|x_2|) - b_1(M_3+M_4), -c_2(|x_1|+|x_2|) + b_1(M_3+M_4),$$
$$c_2(|x_1|+|x_2|) - b_1(M_3+M_4), c_2(|x_1|+|x_2|) + b_1(M_3+M_4)\big)$$

$$M_6 = \max\big(-c_1(|x_3|+|x_4|) - b_2(M_1+M_2), -c_1(|x_3|+|x_4|) + b_2(M_1+M_2),$$
$$c_1(|x_3|+|x_4|) - b_2(M_1+M_2), c_1(|x_3|+|x_4|) + b_2(M_1+M_2)\big)$$

Après simplification, on a :

$$\boxed{M_5 = |c_2|(|x_1|+|x_2|) + |b_1|(M_3+M_4)}$$
$$\boxed{M_6 = |c_1|(|x_3|+|x_4|) + |b_2|(M_1+M_2)}$$

Si on note $A = M_5 + M_6$ et $B = M_9 + M_{10}$, alors on a :

$$x - \varepsilon.M_{11} \le x \le x + \varepsilon.M_{12}$$

où

$$M_{11} = \max\left(\frac{A.\det + n_1.B}{\det(\det + \varepsilon.B)}, \frac{A.\det - n_1.B}{\det(\det - \varepsilon.B)}, \frac{-A.\det + n_1.B}{\det(\det + \varepsilon.B)}, \frac{-A.\det - n_1.B}{\det(\det - \varepsilon.B)}\right)$$

$$M_{12} = \max\left(\frac{-A.\det - n_1.B}{\det(\det + \varepsilon.B)}, \frac{-A.\det + n_1.B}{\det(\det - \varepsilon.B)}, \frac{A.\det - n_1.B}{\det(\det + \varepsilon.B)}, \frac{A.\det + n_1.B}{\det(\det - \varepsilon.B)}\right)$$

Cas du numérateur :
On a A et B positifs, donc le max du numérateur sera : $A.|\det| + |n_1|.B$.

Cas du dénominateur :
On est dans le cas où le déterminant est non nul, donc det et (det ± ε.B) sont de même signe, soit $\det.(\det \pm \varepsilon.B) > 0$.

De plus, $\det - \varepsilon.B < \det + \varepsilon.B$, ce qui implique les inégalités suivantes selon le signe de det :

si $\det > 0 \Rightarrow \det.(\det - \varepsilon.B) < \det.(\det + \varepsilon.B)$

si $\det < 0 \Rightarrow \det.(\det - \varepsilon.B) > \det.(\det + \varepsilon.B)$

d'où

$$\frac{1}{\det(\det + \varepsilon.B)} < \frac{1}{\det(\det - \varepsilon.B)} \qquad \text{si } \det > 0$$

$$\frac{1}{\det(\det - \varepsilon.B)} < \frac{1}{\det(\det + \varepsilon.B)} \qquad \text{si } \det < 0$$

Si on rassemble le numérateur et le dénominateur, on obtient :

$$M_{11} = \frac{A.\det + |n_1|.B}{\det(\det - \varepsilon.B)} \qquad \text{si } \det > 0$$

$$M_{11} = \frac{A.|\det| + |n_1|.B}{\det(\det + \varepsilon.B)} \qquad \text{si } \det < 0$$

On obtient alors $M_{11} = M_{12}$, d'où l'encadrement de x :

$$\mathbf{x - \varepsilon.M_{11} \leq x \leq x + \varepsilon.M_{11}}$$

## ANNEXE 4 : Calcul de l'ordonnée y

### Calcul du numérateur $n_2$ :

De façon similaire au calcul de $n_1$ ( Annexe 3 ), on obtient pour $n_2$ :

$$\boxed{n_2 - \varepsilon(M_7 + M_8) \le n_2 \le n_2 + \varepsilon(M_7 + M_8)}$$

où

$$M_7 = \max\big(-c_1(|y_3|+|y_4|) - a_2(M_1+M_2), -c_1(|y_3|+|y_4|) + a_2(M_1+M_2),$$
$$c_1(|y_3|+|y_4|) - a_2(M_1+M_2), c_1(|y_3|+|y_4|) + a_2(M_1+M_2)\big)$$

$$M_8 = \max\big(-c_2(|y_1|+|y_2|) - a_1(M_3+M_4), -c_2(|y_1|+|y_2|) + a_1(M_3+M_4),$$
$$c_2(|y_1|+|y_2|) - a_1(M_3+M_4), c_2(|y_1|+|y_2|) + a_1(M_3+M_4)\big)$$

Après simplfaction :

$$\boxed{M_7 = |c_1|(|y_3|+|y_4|) + |a_2|(M_1+M_2)}$$
$$\boxed{M_8 = |c_2|(|y_1|+|y_2|) + |a_1|(M_3+M_4)}$$

En posant $C = M_7 + M_8$, on obtient

$$y - \varepsilon.M_{12} \le y \le y + \varepsilon.M_{13}$$

où

$$M_{12} = \max\left(\frac{C.\det + n_2.B}{\det(\det + \varepsilon.B)}, \frac{C.\det - n_2.B}{\det(\det - \varepsilon.B)}, \frac{-C.\det + n_2.B}{\det(\det + \varepsilon.B)}, \frac{-C.\det - n_2.B}{\det(\det - \varepsilon.B)}\right)$$

$$M_{13} = \max\left(\frac{-C.\det - n_2.B}{\det(\det + \varepsilon.B)}, \frac{-C.\det + n_2.B}{\det(\det - \varepsilon.B)}, \frac{C.\det - n_2.B}{\det(\det + \varepsilon.B)}, \frac{C.\det + n_2.B}{\det(\det - \varepsilon.B)}\right)$$

Par la même méthode de simplification pour le calcul de x, on obtient $M_{12} = M_{13}$ et

$$\boxed{M_{12} = \frac{C.\det + |n_2|.B}{\det(\det - \varepsilon.B)} \qquad \text{si } \det > 0}$$

$$\boxed{M_{12} = \frac{C.|\det| + |n_2|.B}{\det(\det + \varepsilon.B)} \qquad \text{si } \det < 0}$$

$$\boxed{\mathbf{y - \varepsilon.M_{12} \le y \le y + \varepsilon.M_{12}}}$$

## ANNEXE 5 : Calcul du déterminant

On a : $det = a_1b_2 - a_2b_1$

D'où les encadrements suivants :

$$a_1b_2 - \varepsilon.M_9 \leq a_1b_2 \leq a_1b_2 + \varepsilon.M_9$$
$$a_2b_1 - \varepsilon.M_{10} \leq a_2b_1 \leq a_2b_1 + \varepsilon.M_{10}$$

où

$$M_9 = \max\left(-a_1(|x_3|+|x_4|) - b_2(|y_1|+|y_2|), -a_1(|x_3|+|x_4|) + b_2(|y_1|+|y_2|),\right.$$
$$\left. a_1(|x_3|+|x_4|) - b_2(|y_1|+|y_2|), a_1(|x_3|+|x_4|) + b_2(|y_1|+|y_2|)\right)$$

$$M_{10} = \max\left(-a_2(|x_1|+|x_2|) - b_1(|y_3|+|y_4|), -a_2(|x_1|+|x_2|) + b_1(|y_3|+|y_4|),\right.$$
$$\left. a_2(|x_1|+|x_2|) - b_1(|y_3|+|y_4|), a_2(|x_1|+|x_2|) + b_1(|y_3|+|y_4|)\right)$$

Après simplification, on a :

$$\boxed{M_9 = 2.(|x_3|+|x_4|)(|y_1|+|y_2|)}$$
$$\boxed{M_{10} = 2.(|x_1|+|x_2|)(|y_3|+|y_4|)}$$

d'où l'encadrement du déterminant :

$$\boxed{\mathbf{det} - \boldsymbol{\varepsilon}.(\mathbf{M_9} + \mathbf{M_{10}}) \leq \mathbf{det} \leq \mathbf{det} + \boldsymbol{\varepsilon}.(\mathbf{M_9} + \mathbf{M_{10}})}$$

## ANNEXE 6 : Calcul de $d_1$

on a :

$$x - \varepsilon M_{11} \le x \le x + \varepsilon M_{11}$$
$$y - \varepsilon M_{12} \le y \le y + \varepsilon M_{12}$$
$$x_1 - \varepsilon|x_1| \le x_1 \le x_1 + \varepsilon|x_1|$$
$$y_1 - \varepsilon|y_1| \le y_1 \le y_1 + \varepsilon|y_1|$$

soit

$$(x - x_1) - \varepsilon(M_{11} + |x_1|) \le x - x_1 \le (x - x_1) + \varepsilon(M_{11} + |x_1|)$$
$$(y - y_1) - \varepsilon(M_{12} + |y_1|) \le y - y_1 \le (y - y_1) + \varepsilon(M_{12} + |y_1|)$$

ce qui nous donne :

$$(x - x_1)^2 - \varepsilon M_{13} \le (x - x_1)^2 \le (x - x_1)^2 + \varepsilon M_{13}$$
$$(y - y_1)^2 - \varepsilon M_{14} \le (y - y_1)^2 \le (y - y_1)^2 + \varepsilon M_{14}$$

où

$$M_{13} = \max(-2(x - x_1)(M_{11} + |x_1|); 2(x - x_1)(M_{11} + |x_1|))$$
$$M_{14} = \max(-2(y - y_1)(M_{12} + |y_1|); 2(y - y_1)(M_{12} + |y_1|))$$

soit,

$$\boxed{M_{13} = 2|x - x_1|(M_{11} + |x_1|)}$$
$$\boxed{M_{14} = 2(|y - y_1|)(M_{12} + |y_1|)}$$

$$\Rightarrow A \le (x - x_1)^2 + (y - y_1)^2 \le B$$

avec $A = (x - x_1)^2 + (y - y_1)^2 - \varepsilon (M_{13} + M_{14})$

$B = (x - x_1)^2 + (y - y_1)^2 + \varepsilon (M_{13} + M_{14})$

Si $A \le 0 \le B$

alors $d_1 = 0$

$A = B = 0$

sinon $\sqrt{A} \le d_1 \le \sqrt{B}$

On obtient donc dans le cas $d_1 \neq 0$

$$\boxed{\sqrt{(x - x_1)^2 + (y - y_1)^2 - \varepsilon(M_{13} + M_{14})} \le d_1 \le \sqrt{(x - x_1)^2 + (y - y_1)^2 + \varepsilon(M_{13} + M_{14})}}$$

## ANNEXE 7 : Algorithmes d'intersection de droites utilisant les intervalles

### Intersection de deux segments

début *{ interseg }*

si les boîtes englobantes ne se coupent pas alors pas d'intersection, int ← 0

sinon *{ calcul des coefficients des deux droites, du déterminant, de $M_9$ et $M_{10}$ }*

si abs(det) < 2.ε.($M_9$+$M_{10}$) alors pas d'intersection, int ← 0

sinon les droites se coupent, int ← 1

*{ calcul de $M_1$,...,$M_8$,$M_{11}$,...,$M_{12}$ }*

$n_1 \leftarrow c_2.b_1 - c_1.b_2$, $en_1 \leftarrow \varepsilon.(M_5+M_6)$

si abs($n_1$) < $en_1$ alors $n_1 \leftarrow 0$

$n_2 \leftarrow c_1.a_2 - c_2.a_1$, $en_2 \leftarrow \varepsilon.(M_7+M_8)$

si abs($n_2$) < $en_2$ alors $n_2 \leftarrow 0$

*{ calcul de x, y, $e_x$, $e_y$ }*

si le point appartient aux deux segments alors int ← 1

sinon int ← 0

si on n'a pas déterminé d'intersection

alors on regarde si les segments se superposent *{ voir II.2 }*

fin

### Appartenance d'un point à un segment

début *{ appar_seg }*

*{ calcul de $M_{13}$, $M_{14}$, A, B pour la distance $d_1$ }*

si ( (A <= 0) et (0 >= B) ) alors

$d_1 \leftarrow 0$, $A \leftarrow 0$, $B \leftarrow 0$, $ed_1 \leftarrow 0$

sinon

$$d_1 \leftarrow \sqrt{(x-x_1)^2+(y-y_1)^2}$$

$$ed_1 \leftarrow (\sqrt{B}-\sqrt{A})/2$$

fsi

*{ calcul de $M_{15}$, $M_{16}$, C, D pour la distance $d_2$ }*

si ( (C <= 0) et (0 >= D) ) alors

$d_2 \leftarrow 0$, $C \leftarrow 0$, $D \leftarrow 0$, $ed_2 \leftarrow 0$

sinon

$$d_2 \leftarrow \sqrt{(x-x_2)^2+(y-y_2)^2}$$

$$ed_2 \leftarrow (\sqrt{D}-\sqrt{C})/2$$

fsi

*{ calcul de $M_{17}$, $M_{18}$, E, F pour la distance $d_3$ }*

si ( (E <= 0) et (0 >= F) ) alors

$d_3 \leftarrow 0,\ E \leftarrow 0,\ F \leftarrow 0,\ ed_3 \leftarrow 0$

sinon

$d_3 \leftarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

$ed_3 \leftarrow (\sqrt{E} - \sqrt{F}) / 2$

fsi

si ( abs($d_1$+$d_2$-$d_3$) <= $ed_1$+$ed_2$+$ed_3$ ) alors appar_seg $\leftarrow$ 1

sinon appar_seg $\leftarrow$ 0

si ( $d_1$ <= $ed_1$ ) alors appar_seg $\leftarrow$ 2

si ( $d_2$ <= $ed_2$ ) alors appar_seg $\leftarrow$ 3

fin

**ANNEXE 8 : Théorème 1**

*Si R est le résultat informatique d'une procédure informatique P finie, ne faisant intervenir que les quatre opérations élémentaires, et, si r représente le résultat obtenu en effectuant la même suite de calculs mais avec une précision infinie, alors en ne retenant que les termes du premier ordre en $2^{-t}$ :*

$$R = r + \sum_{i=1}^{s_n} g_i(d) 2^{E_i - t} \delta_i \alpha_i + O(2^{-2t})$$

*$g_i$ (d) représente des constantes ne dépendant que des données et de l'algorithme,*
*$E_i$ les exposants des résultats intermédiaires,*
*$\alpha_i$ les quantités perdues lors de la troncature ou de l'arrondi*
*$\delta_i$ les signes des résultats intermédiaires,*
*n le nombre d'opérations élémentaires effectuées par l'ordinateur et d'affectation des données.*

Démonstration par récurrence:

Pour n = 1, le résultat est démontré au paragraphe 2. du chapitre IV.

On suppose le résultat vrai pour $k \leq n-1$. Un calcul suivant peut s'interpréter comme le résultat d'une seule opération élémentaire (la dernière effectuée) dont les deux opérandes sont des résultats obtenus au bout d'au plus n-1 opérations.

Soit $\omega$ un opérateur arithmétique exact et $\underline{\underline{\omega}}$ l'opérateur informatique correspondant.

Soit :

$$R = R_1 \underline{\underline{\omega}} R_2 = R_1 \omega R_2 - 2^{E_n - t} \delta_n \alpha_n$$

avec

$$R_j = r_j + \sum_{i=1}^{s_j} g_i^j(d) 2^{E_i^j - t} \delta_i^j \alpha_i^j + O\left(2^{-2t}\right) \quad \text{pour } j = 1,2$$

d'où au premier ordre :

$$R = r_1 \omega r_2 + \left( \sum_{s=1}^{s_n} k_s(d) 2^{E_s - t} \delta_s \alpha_s \right) - 2^{E_n - t} \delta_n \alpha_n + O\left(2^{-2t}\right)$$

avec dans le cas de :

l'addition : $k_s(d) = g_i^j(d)$

la soustraction : $k_s(d) = g_i^1(d)$ ou $-g_i^2(d)$

la multiplication : $k_s(d) = r_2 g_i^1(d)$ ou $-r_1 g_i^2(d)$

la division : $k_s(d) = g_i^1(d) / r_2$ ou $-r_1 g_i^2(d) / r_2^2$

De plus, $k_s(d)$ ne dépend encore que des données et de l'algorithme. Il n'en est pas de même des $E_s$ et $\delta_s$ qui dépendent des arrondis $\alpha_i$ antérieurs et donc de l'arithmétique de l'ordinateur. Ce qui démontre le théorème.

## ANNEXE 9 : Algorithmes de la méthode CESTAC

Le point important de la méthode CESTAC étant la perturbation aléatoire, le logiciel doit permettre :

- de créer une arithmétique aléatoire, c'est à dire d'obtenir comme résultat de toute opération à virgule flottante une valeur aléatoirement par excès ou par défaut,
- de calculer à partir de 2 ou 3 résultats $R_i$ leur moyenne et l'écart type, afin d'en estimer le nombre de chiffres décimaux significatifs.

Certains algorithmes sont spécifiques à la machine utilisée, en particulier, la fonction qui modifie le dernier bit de la mantisse, nous allons développer ces différents programmes pour le matériel utilisé au laboratoire, soit des stations SUN (Sparc station10 et Sparc station classic), à l'aide du langage C et avec une arithmétique arrondie au plus près.

### - FONCTION DERNIER_BIT

Cette fonction permet d'ajouter ou de retrancher 1 au dernier bit de la mantisse d'un réel X. Il existe cependant un problème, celui du zéro exact. Devons-nous modifier le zéro exact?

L'obtention d'un zéro exact au cours d'un calcul en virgule flottante a une probabilité faible de se produire. Les zéro exact qui se trouvent au cours des calculs proviennent souvent, soient des données exactes, soient de différences de deux données identiques au bit près.

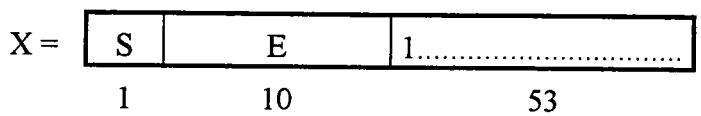
De plus, lorsque nous ajoutons ou retranchons 1 au denier bit de la mantisse de zéro, nous obtenons respectivement les résultats suivant : 4.940656..E-324 et 3.237859..E-319. Vu la taille des exposants, le risque d'obtenir un dépassement de capacité au cours des calculs est très élevé.

Cette dernière remarque nous incite à ne pas modifier le zéro exact. En effet, il semble plus dangereux d'introduire un nombre avec un exposant très faible que de laisser le zéro exact.

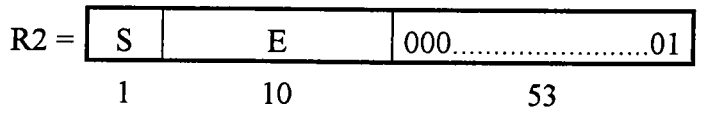
Nous présentons ici le principe pour ajouter ou retrancher 1. On considère les réels de type double, c'est à dire codés sur 8 octets, soit 64 bits. Ils sont répartis de la manière suivante :

- 1 bit pour le signe
- 10 bits pour coder l'exposant biaisé signé
- 53 bits pour la mantisse

La valeur de X se présente donc sous la forme :




Pour obtenir $X^+$ ou $X^-$, il faut additionner ou soustraire à X la valeur 1 au dernier bit de la mantisse. Il faut donc additionner (ou soustraire) à X la valeur R2 suivante :




La valeur R2 est non normalisée, elle a le même signe et le même exposant que X, et une mantisse constituée de 52 zéros et d'un 1 en dernière position. Pour effectuer l'addition (ou la soustraction) de X et R2, on utilise la structure union qui permet de ranger des données dans les mêmes adresses mémoires. Cette union contient le réel codé en double et une structure correspondant à la représentation machine :

```
union reel { double d;
             Enregistrement
               { unsigned int sign : 1;   { bit du signe de d }
                 unsigned int expo : 10;  { 10 bits de l'exposant de d }
                 unsigned int e1   : 21;  { 21 premiers bits de la mantisse }
                 unsigned int e2   : 16;  { 16 bits suivants de la mantisse }
                 unsigned int e3   : 16;  { 16 derniers bits de la mantisse }
               } oct;
             Finenregistrement
           };
```

La décomposition de la mantisse de 53 bits en trois nous permet de gérer la retenue de l'addition ou de la soustraction.

**Addition ou soustraction au dernier bit de la mantisse :**

entrée : X de type double, réel à modifier
S de type int, si S > 0 on ajoute 1, si S < 0 on retranche 1
sortie : la fonction dernier_bit de type double correspondant à $X^+$ ou à $X^-$ contenu dans rep
début *{ dernier_bit }*
si ( X ≠ 0) alors *{ on effectue la perturbation en utilisant des variables }*
*{ locales XR et rep de type reel }*
XR.d ← X; rep.oct.sign ← XR.oct.sign; rep.oct.expo ← XR.oct.expo;
si (S > 0) alors
on ajoute 1 à XR.oct.e3, on met la réponse dans rep.oct.e3 et on gère la retenue
sinon
on retranche 1 à XR.oct.e3, on met la réponse dans rep.oct.e3 et on gère la retenue
fsi
sinon rep.d ← X
fsi
fin

**FONCTION P**

La fonction P calcule aléatoirement $X^+$ ou $X^-$. L'obtention des valeurs aléatoires −1, 0 et +1, nécessaires à ce calcul, avec les probabilités p suivantes : p(−1) = p(+1) = 0.25 et p(0) = 0.5 pour l'arithmétique arrondie au plus près, sont obtenus en prenant la partie entière de la fonction hasard appelée avec alpha = -1.999...99 et béta = 1.999...99.

**Calcul de $X^+$ ou $X^-$, le nombre Perturbé :**

entrée : X de type double, réel a perturbé

sortie : la fonction P de type double correspondant à $X^+$ ou $X^-$ contenu dans rep

```
début   { P }
   si ( X = 0) alors  rep ← 0
   sinon     pile ← hasard(-1.99999999999999, 1.99999999999999)
             si (pile = 0) alors rep ← X
             sinon rep ← dernier_bit(X,pile)
             fsi
   fsi
fin
```

Remarque : La fonction hasard renvoie un nombre aléatoire entre les deux bornes passées en paramètres.

**PROCÉDURE DE CALCUL DU NOMBRE DE CHIFFRES DECIMAUX SIGNIFICATIFS D'UN RÉSULTAT (NCSE)**

Cette procédure calcule, à partir de trois résultats $R_i$, la moyenne $\overline{R}$, l'écart type S et le nombre de chiffres décimaux significatifs $C_{\overline{R}}$ (§ 4.2.5.). Bien entendu, les cas particuliers doivent être considérés tels : S = 0 ou $\overline{R}$ = 0.

Ici, la précision maximale de la machine a été prise égale à 16 chiffres décimaux, les valeurs $R_i$ sont rangées dans un tableau R, la moyenne est représentée par RM et $C_{\overline{R}}$ par C.
Nous affectons la valeur 16 à C dans deux cas :

- lorsque l'écart-type est égal à zéro, c'est-à-dire, que malgré les perturbations, nous avons obtenons trois fois le même résultat.
- lorsque le nombre de chiffres significatifs est supérieur à la précision machine, à savoir, quand C > 16.

Si le nombre de chiffres significatifs calculé est négatif, il correspond forcément à 0 chiffre significatif. Or $\log_{10}(g) - 0{,}39$ devient négatif pour g < 2,5, donc nous mettons C à zéro dès que g < 2,5.

**Calcul du Nombre de Chiffres Significatifs Estimés** :

entrée : R tableau des trois résultats de type double

sortie : RM de type double, moyenne des trois échantillons
C de type double, nombre de chiffres significatifs estimés

```
début   { NCSE }
   RM ← (R[0] + R[1] + R[2])/3;
   var ← (R[0] – RM)^2 + (R[1] – RM)^2 + (R[2] – RM)^2 ;
   ect ← sqrt(var/2);
   si (ect ≠ 0) alors
      g ← | RM / ect |;
      si (g > 2,5) alors
         C ← log10(g) – 0.39;
         si (C > 16) alors C ← 16
         fsi
      sinon C ← 0
      fsi
   sinon C ← 16
   fsi
fin
```

**ANNEXE 10 : Théorème 2**

*Si $p_n$ et $q_n$ sont définis par :*

$$\begin{cases} p_0 = a_0 \\ p_1 = a_1 a_0 + 1 \\ p_n = a_n p_{n-1} + p_{n-2} \quad (2 \le n \le N) \end{cases} \quad et \quad \begin{cases} q_0 = 1 \\ q_1 = a_1 \\ q_n = a_n q_{n-1} + q_{n-2} \quad (2 \le n \le N) \end{cases}$$

*alors*

$$[a_0, a_1, \ldots, a_n] = \frac{p_n}{q_n}.$$

Démonstration : (par récurrence)

Pour n = 0 :

$$[a_0] = \frac{a_0}{1} = \frac{p_0}{q_0}$$

Pour n = 1 :

$$[a_0, a_1] = a_0 + \frac{1}{a_1} = \frac{a_0 a_1 + 1}{a_1} = \frac{p_1}{q_1}$$

Supposons la propriété vrai jusqu'au rang m (< N), on a :

$$[a_0, a_1, \ldots, a_m] = \frac{p_m}{q_m} = \frac{a_m p_{m-1} + p_{m-2}}{a_m q_{m-1} + q_{m-2}}$$

Montrons que :

$$[a_0, a_1, \ldots, a_{m+1}] = \frac{p_{m+1}}{q_{m+1}} = \frac{a_{m+1} p_m + p_{m-1}}{a_{m+1} q_m + q_{m-1}}$$

$$\begin{aligned} [a_0, a_1, \ldots, a_m, a_{m+1}] &= [a_0, a_1, \ldots, a_{m-1}, [a_m, a_{m+1}]] \\ &= \left[ a_0, \ldots, a_{m-1}, a_m + \frac{1}{a_{m+1}} \right] \\ &= \frac{(a_m a_{m+1} + 1) p_{m-1} + a_{m+1} p_{m-2}}{(a_m a_{m+1} + 1) q_{m-1} + a_{m+1} q_{m-2}} \\ &= \frac{a_{m+1} p_m + p_{m-1}}{a_{m+1} q_m + q_{m-1}} \end{aligned}$$

$$[a_0, a_1, \ldots, a_{m+1}] = \frac{p_{m+1}}{q_{m+1}}$$

$$\begin{aligned} &= \frac{(a_m a_{m+1} + 1) p_{m-1} + a_{m+1} p_{m-2}}{(a_m a_{m+1} + 1) q_{m-1} + a_{m+1} q_{m-2}} \\ &= \frac{a_{m+1}(a_m p_{m-1} + p_{m-2}) + p_{m-1}}{a_{m+1}(a_m q_{m-1} + q_{m-2}) + q_{m-1}} \\ &= \frac{a_{m+1} p_m + p_{m-1}}{a_{m+1} q_m + q_{m-1}} \end{aligned}$$

$[a_0, a_1, \ldots, a_{m+1}] = \frac{p_{m+1}}{q_{m+1}}$. Ce qui démontre le théorème.

## ANNEXE 11 : Théorème 3

*Tout nombre irrationnel peut être exprimé d'une façon unique par une fraction continue simple infinie.*

Démonstration :

Montrons que si $\alpha \notin \mathbf{Q}$, alors il existe une unique suite $a_0, a_1,...$ telle que $\alpha = \lim_{n\to\infty} \frac{p_n}{q_n}$.

Si on construit la suite $a_0, a_1,...$ de la façon suivante :

$a_0 = [\alpha] \quad \alpha_1 = \frac{1}{\alpha - a_0} > 1$ ( $[\alpha]$ : partie entière de $\alpha$)

$a_1 = [\alpha_1] \quad \alpha_2 = \frac{1}{\alpha_1 - a_1} > 1$

etc ...

$a_k = [\alpha_k] \quad \alpha_{k+1} = \alpha_{k+1} = \frac{1}{\alpha_k - a_k}$.

Soit $\beta = \lim_{n\to\infty} \frac{p_n}{q_n}$, montrons que $\alpha = \beta$.

Par construction, $\alpha = [a_0, a_1, ...,a_n, \alpha_{n+1}]$, et on a :

$$\begin{aligned}
\alpha - \frac{p_n}{q_n} &= \frac{\alpha_{n+1}p_n + p_{n-1}}{\alpha_{n+1}q_n + q_{n-1}} - \frac{p_n}{q_n} \\
&= \frac{(\alpha_{n+1}p_n + p_{n-1})q_n - p_n(\alpha_{n+1}q_n + q_{n-1})}{(\alpha_{n+1}q_n + q_{n-1})q_n} \\
&= \frac{\alpha_{n+1}p_nq_n + p_{n-1}q_n - p_n\alpha_{n+1}q_n - p_nq_{n-1}}{(\alpha_{n+1}q_n + q_{n-1})q_n} \\
&= \frac{-(p_nq_{n-1} - p_{n-1}q_n)}{(\alpha_{n+1}q_n + q_{n-1})q_n} \\
&= \frac{(-1)^n}{(\alpha_{n+1}q_n + q_{n-1})q_n}
\end{aligned}$$

Donc :

$\left|\alpha - \frac{p_n}{q_n}\right| < \frac{1}{q_n^2}$, ce qui implique que la suite $\left(\frac{p_n}{q_n}\right)$ converge vers $\alpha$, donc $\alpha = \beta$.

Tout nombre irrationnel peut alors être représenté par une fraction continue infinie simple.

Montrons l'unicité :

Supposons qu'il existe deux fractions continues infinies simples, $[a_0, a_1,...]$ et $[b_0, b_1,...]$, représentant le même nombre irrationnel $\alpha$. Montrons alors que $a_k = b_k$ pour $k = 1, 2, ...$ .

On a $[a_0, a_1,...] = [b_0, b_1,...]$

$$\alpha = [a_0, a_1, \ldots] = \lim_{n \to \infty} [a_0, a_1, \ldots, a_n]$$

On a $0 \leq \alpha - a_0 < 1$ donc $a_0 = [\alpha]$, de même $b_0 = [\alpha]$

Ce qui nous donne $a_0 = b_0$ et $a_0 + \frac{1}{[a_1, a_2, \ldots]} = b_0 + \frac{1}{[b_1, b_2, \ldots]}$

Donc $[a_1, a_2, \ldots] = [b_1, b_2, \ldots]$

Supponsons la propriété vraie jusqu'à k, i.e. $[a_{k+1}, a_{k+2}, \ldots] = [b_{k+1}, b_{k+2}, \ldots]$, on obtient comme précédemment, que $a_{k+1} = b_{k+1}$ et $a_{k+1} + \frac{1}{[a_{k+2}, a_{k+3}, \ldots]} = b_{k+1} + \frac{1}{[b_{k+2}, b_{k+3}, \ldots]}$

Donc $[a_{k+2}, a_{k+3}, \ldots] = [b_{k+2}, b_{k+3}, \ldots]$

Et par suite, on obtient $a_k = b_k$ pour $k = 1, 2, \ldots$ .
Un nombre irrationnel peut donc être exprimé d'une façon unique par une fraction continue simple infinie.

## ANNEXE 12 : Algorithmes de la méthode E.C.P.

**Conservation du rationnel GEN dans le réel :**

entrée : deux entiers GEN gp, gq correspondant respectivement au numérateur et dénominateur

sortie : le réel w dans lequel nous sauvegardons le rationnel

```
début   { gen_ration }
   w.num[0] ← lgef(gp);          { sauvegarde de la longueur effective de l'entier GEN gp }
   w.den[0] ← lgef(gq);          { sauvegarde de la longueur effective de l'entier GEN gq }
   w.s ← signe(gp)*signe(gq);    { sauvegarde du signe du rationnel gp/gq }
   pour i ← 1 à la longueur lgef(gp) – 1 faire
      w.num[i] ← mant(gp,i);     { sauvegarde de chaque mantisse des mots de l'entier gp }
   fpour
   pour i ← 1 à la longueur lgef(gq) – 1 faire
      w.den[i] ← mant(gq,i);     { sauvegarde de chaque mantisse des mots de l'entier gq }
   fpour
fin
```

**Transformation du numérateur du réel (conservé dans num) en un entier GEN :**

entrée : le réel w

sortie : la fonction entier_gen1 de type GEN, qui contient l'entier GEN correspondant au numérateur du rationnel approchant le réel w

```
début   {entier_gen1 }
   si w.den[0] = 0 alors          { le rationnel correspondant n'a pas été calculé }
      calcul du rationnel correspondant au nombre flottant w.x   { appel de convert }
   fsi
   l1 ← w.num[0];                 { récupération de la longueur effective }
   gr ← cgeti(2+l1);              { création d'un entier GEN }
   setsigne(gr,w.s);              { précision du signe du rationnel }
   setlgef(gr,l1);                { précision de la longueur effective de l'entier }
   pour i ← 1 à l1 – 1 faire
      setmant(gr,i,w.num[i]);     { récupération des mantisses qui forment l'entier GEN }
   fpour
   renvoi de gr;
fin
```

**Transformation du dénominateur du réel (conservé dans den) en un entier GEN :**

entrée : le réel w

sortie : la fonction entier_gen2 de type GEN, qui contient l'entier GEN correspondant au dénominateur du rationnel approchant le réel w

```
début   { entier_gen2 }
   si w.den[0] = 0 alors          { le rationnel correspondant n'a pas été calculé }
      calcul du rationnel correspondant au nombre flottant w.x   { appel de convert }
   fsi
   l1 ← w.den[0];                 { récupération de la longueur effective }
   gr ← cgeti(2+l1);              { création d'un entier GEN }
   setsigne(gr,w.1);              { signe du dénominateur est imposé positif (=1) }
   setlgef(gr,l1);                { précision de la longueur effective de l'entier }
   pour i ← 1 à l1 − 1 faire
      setmant(gr,i,w.den[i]);     { récupération des mantisses qui forment l'entier GEN }
   fpour
   renvoi de gr;
fin
```

## ANNEXE 13 :

### Algorithme du test de nullité

entrée : les données $d_i$ de type "réel" servant aux calculs

```
début
   pour i de 1 à 3 faire
      perturber les données d_i
      calcul des trois valeurs de chaque opérande
      calcul de RM et de CRM
   fpour
   calcul des moyennes om1, om2 des opérandes et de leur chiffres significatifs
   si ( com1 < 1 et com2 < 1 ) alors
      utilisation de l'arithmétique exacte  { troisième cas }
   sinon
      calcul de RM et CRM
      si ( CRM < 1 ou RM = 0) alors     { cas du zéro informatique ou }
         instruction 1                  { du zéro exact avec une valeur significative }
      sinon
         instruction 2                  { valeur significative et RM ≠ 0 }
      fsi
   fsi
fin
```

### Algorithme du test d'égalité

entrée : a,b deux variables de type "réel"
sortie : la fonction égale de type booléen, qui renvoie vrai si a = b, faux sinon

```
début   { égale }
   pour i de 1 à 3 faire
      perturber a et b
      calculer la différence des valeurs perturbées, ce qui nous donne un tableau d
   fpour
   calcul de la moyenne dm des trois valeurs et du nombre de chiffres significatifs cdm
   si cdm < 1 ou dm = 0 alors  renvoyer vrai
   sinon renvoyer faux
   fsi
fin
```

**Algorithme du test d'infériorité**

entrée : a,b deux variables de type "réel"
sortie : la fonction inf de type booléen, qui renvoie vrai si a < b, faux sinon

```
début   { inf }
   pour i de 1 à 3 faire
      perturber a et b
      calculer la différence des valeurs perturbées, ce qui nous donne un tableau d
   fpour
   calcul de la moyenne dm des trois valeurs et du nombre de chiffres significatifs cdm
   si cdm < 1 ou dm > 0 alors
      renvoyer faux
   sinon renvoyer vrai
   fsi
fin
```

**Algorithme du test de supériorité**

entrée : a,b deux variables de type "réel"
sortie : la fonction sup de type booléen, qui renvoie vrai si a > b, faux sinon

```
début   { sup }
   pour i de 1 à 3 faire
      perturber a et b
      calculer la différence des valeurs perturbées, ce qui nous donne un tableau d
   fpour
   calcul de la moyenne dm des trois valeurs et du nombre de chiffres significatifs cdm
   si cdm < 1 ou dm < 0 alors
      renvoyer faux
   sinon renvoyer vrai
   fsi
fin
```

# RÉFÉRENCES BIBLIOGRAPHIQUES

## RÉFÉRENCES BIBLIOGRAPHIQUES

[ALI 85] Alliot N., "Logiciels d'identification de paramètres dans des modèles non linéaires avec estimation de la précision des résultats", thèse, Université Pierre et Marie Curie, Paris VI, novembre 1985.

[AQU 96] Aquarels product WW Home Page, www.simulog.fr/aquarels/WWW/aquarels/HomePage.html

[BCO 95] Batut C., Bernardi D., Cohen H., Olivier M., "User's Guide to PARI-GP". Laboratoire A2X, Université Bordeaux I, 351 Cours de la Libération, 33405 Talence Cedex, France.
e-mail : pari@math.u-bordeaux.fr, ftp: megrez.math.u-bordeaux.fr.

[BEN 93] Benouamer M.O., "Opérations booléennes sur les polyèdres représentés par leurs frontières et imprécisions numériques", thèse, Ecole Nationale Supérieure des Mines de St Etienne, juillet 1993.

[BER 95] Bernard P., "AQUARELS - A Problem-Solving Environment for Numerical Quality", Computational Mathematics, ERCIM News n° 22, July 1995, INRIA.

[BRE 88] BrezinskiC., "Algorithme numérique", ellipses 1988;

[CHA 88] Chatelin F., "Sur le taux de fiabilité général de la méthode CESTAC", Comptes rendus de l'Académie des sciences, tome 307 série I, 851-854, juin 1988.

[CHE 88] Chesneaux J.M., "Modélisation et conditions de validité de la méthode CESTAC", Comptes rendus de l'Académie des sciences, tome 307 série I n°1, 417-422, juin 1988.

Chesnaux J.M., Vignes J., "Sur la robustesse de la méthode CESTAC", Comptes rendus de l'Académie des sciences, tome 307 série I n°1, 855-860, 1988.

Chesnaux J.M., "Étude théorique et implémentation en ADA de la méthode CESTAC", thèse, Université de Paris VI, 1988.

[CIA 82] Ciarlet P.G., "Introduction à l'analyse numérique matricielle et à l'optimisation", Masson 1982.

[DAN 88] Daniel M., "Étude des algorithmes de calcul d'un point d'une courbe ou surface B-splines", Revue Internationale de CFAO et d'Infographie, vol 3, n°4, 33-56, 1988.

[DAN 89] Daniel M., "Modélisation de courbes et surfaces par des B-splines. Application à la conception et à la visualisation des formes", thèse, Ecole Nationale Supérieure de Mécanique de Nantes, mai 1989.

[DAU 89] Daubisse J.C., Daniel M., "The numerical problem of using Bézier curves and surfaces in the power basis", Computer Aided Geometric Design 6, 121-128, 1989.

[DEM 91] Demailly J.P., "Analyse numérique et équations différentielles", chapitre 1, presses universitaires de Grenoble 1991.

[DAV 93] Davenport J., Siret Y., Tournier E., "Calcul formel. Systèmes et algorithmes de manipulations algébriques", Masson 1993.

[ERH 93] Erhel J., "Experiments with data perturbations to study condition numbers and numerical stability", Rapport technique, Computing, INRIA/IRISA 1993.

[EYS 96] Eyssette F., Faure C., Gilbert J. C., Rostaing-Schmidt N., " Applicabilité de la différentiation automatique à un système d'équations aux dérivées partielles régissant les phénomènes thermohydroliques dans un tube chauffant.", Rapport de recherche n° 2795, INRIA, février 1996, 96 pages.

[FAL 91] Fall M. M., "Contribution à la détermination de la précision de calcul des algorithmes de traitements numériques", thèse, École centale de Paris, juin 1991.

[FAY 87] Faye J.P., "Approche stochastique de la propagation des erreurs d'arrondis dans les méthodes itératives : application à l'algorithme QR de calcul de valeurs propres", thèse, Université de Paris VI, 1987.

[FAY 88] Faye J.P., "Implémentation synchrone de CESTAC", Comptes rendus de l'Académie des sciences, tome 309 série I, 637-640, juin 1988.

[FEL 76] Feldstein A., Goodman R., "Convergence estimates for the distribution of trailing digits", Journal of the Association for Computing Machinery, volume 23, n° 2, 287-297, avril 1976.

[FEL 86] Feldstein A., Turner P., "Overflow, underflow, and sever loss of significance in floating-point addition and subtraction", IMA Journal of numerical analysis, 6, 241-251, 1986.

[FRA 89] François P., "Contribution à l'étude de méthodes de contrôle de l'erreur d'arrondi : la méthodologie SCALP", thèse, I.N.P Grenoble, 1989.

[GAR 88] Gardan Y., "Éléments de C.A.O. volume 1: Matériels et logiciels de base", Hermès, Paris 1988.

[GAR 94] Gardan Y., "REGAIN : an adaptable CAD system, using a novel man-machine interface and generic applications models", deuxième congrès Franco-Japonais de mécatronique, Takamatsu, Japon, octobre 1994.

[GAP 95] Gardan Y. Pirus D., "Imprécisions numériques en C.A.O. Étude de la propagation des erreurs", rapport de recherche n°95-09 au Laboratoire de Recherche en Informatique de Metz.

[GOL 91] Goldberg D., "What every computer scientist should know about floating-point arithmetic", ACM Computing Surveys, volume 23, n° 1, 5-48, mars 1991.

[GOM 95] Gomez CI. Salvy B. Zimmermann P., "Calcul formel : Mode d'emploi. Exemples en Maple", Masson 1995.

[HAW 60] Hardy G.H. Wright E.M., "An introduction to the theory of numbers", Oxford University Press 4th Ed. 1960.

[HHK 89] Hoffmann M.C., Hopcroft J.E., Karasick M.S., "Robust set operations on polyhedral solids", IEEE Computer Graphics and Applications, vol. 9, 50-59, novembre 1989.

[HOF 88] Hoffmann M.C., Hopcroft J.E., Karasick M.S., "Towards implementation robust geometric computations", 4th ACM Symposium. Computational Geometry, ACM, New York, 106-117, 1988.

[HOF 89] Hoffmann M.C., "Geometric and Solid Modeling, an introduction", chapitre Kaufmann, San Francisco 1989.

Hoffmann M.C., "The problems of accuracy and robustness in geometric computation", IEEE Computer 22, 31-42 1989.

[JAI 93] Jaillon P., "Proposition d'une arithmétique rationnelle paresseuse et d'un outil d'aide à la saisie d'objets en synthèse d'images", thèse, École Nationale Supérieure des mines de Saint-Étienne, septembre 1993.

[KNU 61] Knuth D. E., "The art of computer programming", volume 2, 1961, second edition (1981), Addison-Wesley Publishing Company.

[LUC 61] Lucas E., "Théorie des nombres", tome1, Librairie scientifique et technique Albert Blanchard, 1961.

[MAI 82] Maillé M., "Some methods to estimate accuracy of measurements or numerical computations", Processings of Maths. for Comp. Congress AFCET, Paris 1982.

[MAP 93] Maple V, "Library reference manual", Springer Verlag 1993.

[MAS 93] Masotti G., "Floating-point numbers with error estimates", Computer-Aided Design, 25(9), 524-538, septembre 1993.

[MAT 95] Matlab, "High Performance Numeric Computation and Visualisation Software", External Interface Guide, The Mathworks INC, Avril 1995.

[MIC 87] Michelucci D., "Les représentation par les frontières : quelques constructions; difficultés rencontrées", thèse, Ecole Nationale Supérieure des Mines de St Etienne, novembre 1987.

[MIL 88] Milenkovic V.J., "Verifiable implementations of geometric algorithms using finite precision arithmetic", Artificial Intelligence 37, 377-401, décembre 1988.

[MOR 90] Moreau .J.M., "Hiérarchisation et facetisation de la représentation par segments d'un graphe planaire dans le cadre d'une arithmétique mixte. Thèse, Ecole Nationale Supérieure des Mines de St Etienne, octobre 1990.

[MUL 89] Muller J.M., "Arithmétique de l'ordinateur. Opérateurs et fonctions élémentaires", chapitre 2, Masson 1989.

[PIR 96] Pirus D., "Erreurs numériques-Estimations et contrôle de la précision", Les Quatrièmes Journées de l'Association Française d'Informatique Graphique, AFIG, Dijon, novembre 1996.

[ROS 93] Rosen K.H., "Elementary number theory and its applications", third edition, Addison-Wesley publishing company, 1993.

[SHA 93] Shahbazkia HR., "Apport des logiciels de calcul formel à la modélisation géométrique en C.F.A.O". Rapport de stage de D.E.A d'Informatique réalisé au Laboratoire de Recherche d'Informatique de Metz.

[STE 75] Stewart N. F., Jensen F., "Solutions numériques des problèmes matriciels", Les presses de l'Université de Montréal, Eyrolles Paris.

[STE 93] Stewart N. F., Desaulniers H., "Robustness of numerical methods in geometric computation when problem data is uncertain", Computer Aided Design, volume 25, n° 9, 539-545, septembre 1993

[STO 93] Stoer J., Bulirsch R., "Introduction to Numérical Analysis", chapitre 1 by Springer Verlag N.Y, seconde édition 1993.

[VIG 74] Vignes J., La Porte M., "Etude statistique des erreurs dans l'arithmétique des ordinateurs; application au contrôle des résultats d'algorithmes numériques", Numerische Mathematik 23, 63-72, Springer-Verlag 1974.

[VIG 78] Vignes J., "New methods for evaluating the validity of results of mathematicals computation", Mathematics and Computer in Simulation, 227-249 vol 20 n°4 1978.

[VIG 84] Vignes J., "Implémentation des méthodes d'optimisation : Test d'arrêt optimal, contrôle et précision de la solution (I)", RAIRO, Recherche Opérationnelle, volume 18, n°1, 1-18, février 1984.

[VIG 84] Vignes J., "Implémentation des méthodes d'optimisation : Test d'arrêt optimal, contrôle et précision de la solution (II)", RAIRO, Recherche Opérationnelle, volume 18, n°2, 103-129, mai 1984.

[VIG 86] Vignes J., "Zéro mathématique et zéro informatique", Comptes rendus de l'Académie des sciences, tome 303, série I n°1, 997-1000, juin 1986.

[VIG 87] Vignes J., "Zéro mathématique et zéro informatique", La vie des sciences, CRAS, tome 4, n°1, 1-13, janvier-février 1987.

[VIG 93] Vignes J., Pichat M., "Ingénierie du contrôle de la précision des calculs sur ordinateur", Technip Paris 1993.