Université Pierre et Marie Curie - Paris 6

Mémoire présenté par

Fabienne Jézéquel

en vue d'obtenir l'

Habilitation à diriger des recherches

Contrôle dynamique de méthodes d'approximation

soutenue le 10 février 2005 devant le jury composé de :

Götz Alefeld,	Professeur à Universität Karlsruhe, Allemagne,	Examinateur,
Claude Brezinski,	Professeur à l'Université Lille 1,	Rapporteur,
Jean-Marie Chesneaux,	Professeur à l'Université Paris 6,	Examinateur,
Nicholas J. Higham,	Professeur à University of Manchester, Royaume-Uni,	Examinateur,
Bernard Philippe,	Directeur de recherches à l'INRIA, IRISA, Rennes,	Rapporteur,
Marcos Raydan,	Professeur à Universidad Central de Venezuela,	Rapporteur,
Jean Vignes,	Professeur émérite à l'Université Paris 6,	Président.

Remerciements

Tout d'abord, je tiens à exprimer ma profonde gratitude envers le professeur Jean-Marie Chesneaux. Je le remercie pour la confiance qu'il m'a toujours témoignée et les encouragements qu'il m'a maintes fois prodigués. Je suis très honorée d'avoir pu préparer mon habilitation auprès d'un chercheur d'une telle qualité.

Je souhaite exprimer ma reconnaissance envers le professeur Jean Vignes. Il a toujours montré de l'intérêt pour mon travail, en me faisant bénéficier de sa grande compétence et de son caractère jovial. Je le remercie sincèrement pour ses précieux conseils.

Je remercie vivement le professeur Claude Brezinski, Bernard Philippe et le professeur Marcos Raydan d'avoir accepté la tâche de rapporteur et de me faire l'honneur d'être tous présents à la soutenance. Les remarques émises par le professeur Claude Brezinski lors de la lecture de ce mémoire ont permis d'en améliorer la qualité.

Je suis très reconnaissante envers les professeurs Götz Alefeld et Nick Higham de l'honneur qu'ils m'ont fait en acceptant de faire partie du jury d'habilitation.

Mes activités de recherche se déroulent dans d'excellentes conditions grâce aux membres du Laboratoire d'Informatique de Paris 6 (LIP6). Je souhaite exprimer toute ma gratitude envers le professeur René Alt qui m'a fait l'honneur d'encadrer ma thèse de doctorat. Je le remercie pour ses qualités scientifiques et humaines et pour son soutien constant. Je tiens à souligner l'atmosphère conviviale qui règne dans le thème "Algorithmique Numérique et Parallélisme" du LIP6. J'en remercie tous les membres (anciens et présents) qui m'ont, avec une grande disponibilité, apporté leur aide. Mes remerciements vont en particulier aux doctorants que j'ai co-encadrés avec le professeur Jean-Marie Chesneaux : Mourad Charikhi, dont la soutenance a précédé la mienne de quelques semaines, et Robert Adout, auquel j'adresse mes plus vifs encouragements pour achever sa thèse. Je remercie les membres du thème "Réseaux et Performances" du LIP6, qui ont été mes proches voisins sur le campus Jussieu puis sur le site de Scott, pour la sympathie qu'ils ont toujours témoignée à mon égard. Je remercie les ingénieurs du LIP6 qui ont toujours traité questions et problèmes avec beaucoup d'efficacité. Je n'oublie pas le personnel administratif qui prend en charge, de manière très chaleureuse, les aspects pratiques ou financiers des activités de recherche menées au LIP6. Enfin, je profite de ces lignes pour dire combien j'ai apprécié, depuis mon arrivée à l'université Paris 6, le soutien d'Anne Fladenmuller, Jean-Luc Lamotte et Clémentine Saïd.

Je remercie le CCR (Centre de Calcul et de Recherche) de Jussieu et l'IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) pour leur aide lors de l'utilisation de leurs machines parallèles ou vectorielles, en particulier Jean-Michel Dupays qui m'a toujours informée des évolutions du super-calculateur NEC SX5.

L'université Paris 2, où j'exerce mes activités d'enseignement, est un lieu d'échanges très fructueux. Je remercie mes collègues informaticiens et mathématiciens pour leur gentillesse et leur bonne humeur. Je leur suis très reconnaissante pour leur compréhension et leur soutien lors de la préparation de mon habilitation.

Je souhaite adresser mes encouragements à tous ceux qui envisagent de passer leur habilitation dans mon entourage à Paris 2 ou Paris 6.

Je finirai en associant à ces remerciements ma famille et mes amis pour leur soutien, en particulier mes parents qui accordent beaucoup d'importance à cette habilitation.

Jean-Philippe et Delphine méritent plus que des remerciements. Je leur dédie ce mémoire, avec tout mon amour.

Résumé

Une méthode d'approximation fournit un résultat entaché d'une erreur de méthode inhérente à l'algorithme utilisé et d'une erreur d'arrondi due à la précision finie de l'arithmétique de l'ordinateur. Lorsque le pas de discrétisation d'une telle méthode décroît, l'erreur de méthode diminue, mais l'erreur d'arrondi augmente. Il peut alors être difficile de contrôler ces deux erreurs à la fois. L'Arithmétique Stochastique Discrète (ASD) est une méthode automatique d'analyse d'erreur d'arrondi, qui utilise un mode d'arrondi aléatoire. Son coût doit rester modéré quelle que soit la machine utilisée. Une implémentation de l'ASD a été effectuée sur architecture vectorielle pour permettre l'utilisation du mode d'arrondi aléatoire sans inhiber la vectorisation. Afin d'obtenir avec une méthode d'approximation un résultat pour lequel l'erreur globale (qui comprend l'erreur de méthode et l'erreur d'arrondi) est minimale, une stratégie, fondée sur un calcul de suite, a été proposée. Les calculs sont effectués jusqu'à ce que la différence entre deux itérés successifs soit non significative. Il est alors possible de déterminer quels chiffres du résultat obtenu sont communs avec la solution exacte. Cette stratégie peut s'appliquer au calcul d'intégrales par la méthode des trapèzes, de Simpson, de Romberg ou de Gauss-Legendre. Les différents résultats théoriques qui ont été établis peuvent se combiner si une méthode numérique nécessite le calcul de plusieurs suites. Lorsqu'une intégrale sur un domaine infini est approchée par une somme d'intégrales sur des domaines finis, le nombre optimal d'intervalles traités et le pas d'intégration optimal pour chaque intervalle peuvent être déterminés dynamiquement. Deux approches ont été étudiées pour le contrôle dynamique du calcul d'intégrales multiples. Le principe des "intégrales itérées" consiste à décomposer l'intégrale multiple en plusieurs intégrales monodimensionnelles, qui peuvent être évaluées par une méthode de quadrature. On peut alors déterminer dynamiquement la valeur optimale du pas d'intégration pour chaque dimension du domaine traité. Une intégrale multiple peut aussi être évaluée par une méthode de cubature, adaptée au domaine traité. La stratégie proposée consiste alors à calculer une suite d'approximations fondées sur des divisions successives du domaine d'intégration. Les différents résultats théoriques obtenus ont été appliqués au calcul de deux intégrales impropres : l'une établie en cristallographie, l'autre issue de l'étude des étoiles à neutrons.

Mots-clé : erreur d'arrondi, calcul à haute performance, méthodes d'approximation, intégration numérique, méthodes de quadrature, méthodes de cubature.

Abstract

An approximation method provides a result which is affected by a truncation error inherent to the algorithm used and a round-off error due to the finite precision of the computer arithmetic. If the discretization step of such a method decreases, the truncation error also decreases, but the round-off error increases. Therefore it can be difficult to control these two errors simultaneously. Discrete Stochastic Arithmetic (DSA) is an automatic method for round-off error analysis, which uses a random rounding mode. Its computing cost should remain moderate whatever the machine used. DSA has been implemented on vector computer architectures to allow the use of the random rounding mode without inhibiting vectorization. In order to obtain with an approximation method a result for which the global error (consisting of both the truncation error and the round-off error) is minimal, a strategy, based on a converging sequence computation, has been proposed. Computation is carried out until the difference between two successive iterates has no exact significant digit. Then it is possible to determine which digits of the result obtained are in common with the exact solution. This strategy can apply to the computation of integrals using the trapezoidal rule, Simpson's rule, Romberg's method or the Gauss-Legendre method. The different theoretical results which have been established can combine if a numerical method requires the computation of several sequences. If an integral on an infinite domain is approximated by a sum of integrals on finite domains, one can dynamically determine both the optimal number of intervals and the optimal integration step on each interval. A dynamical control of multiple integrals computation has been proposed using two different approaches. The principle of "iterated integrals" consists in expressing a multiple integral as several one-dimensional integrals, which can be evaluated using a quadrature method. Thus one can dynamically determine the optimal integration step for each dimension of the domain considered. A multiple integral can also be evaluated using a cubature method, adapted to the integration domain. The strategy which has been proposed consists in computing a sequence of approximations based on successive partitions of the integration domain. The different theoretical results which have been obtained have been applied to the computation of two improper integrals: one involved in crystallography, the other one in the neutron star theory.

Key-words : round-off error, high performance computing, approximation methods, numerical integration, quadrature methods, cubature methods.

Table des matières

Chapitre 1

Introduction

1.1 Simulation numérique et précision finie

La simulation numérique s'est considérablement développée depuis l'avènement des ordinateurs au milieu du vingtième siècle. Elle est utilisée dans des domaines aussi divers que l'aéronautique, la géophysique, la climatologie ou l'astrophysique afin de traiter des problèmes pour lesquels il n'existe pas de solution analytique simple. Elle permet, en outre, de s'affranchir d'expérimentations coûteuses (essais en soufflerie ou *crash tests* pour la conception de véhicules), voire nocives pour l'environnement dans le cas notamment des essais nucléaires.

Dans toute démarche scientifique aboutissant à une simulation numérique, interviennent différents types d'erreur. Tout d'abord, il existe une incertitude sur les données mesurées au niveau physique. De plus, lors du choix de l'équation décrivant le phénomène physique, peut se produire une erreur de modélisation. L'utilisation d'une méthode numérique impose le passage d'un modèle continu à sa représentation discrète. Une erreur de troncature (ou erreur de méthode) est alors commise. Celle-ci, intrinsèque à la méthode choisie, dépend de la valeur du pas de discrétisation utilisé. Enfin la précision finie de l'arithmétique de l'ordinateur génère une erreur d'arrondi. Celle-ci survient en général à chaque affectation de variable et à chaque opération arithmétique.

La norme IEEE 754 de 1985 [86] spécifie les caractéristiques de l'arithmétique à virgule flottante des ordinateurs, notamment la taille de la mantisse et de l'exposant. Son apparition a permis d'améliorer la qualité numérique des résultats et la portabilité des programmes. En portant leurs codes d'un calculateur avec une arithmétique spécifique vers une machine respectant la norme IEEE 754, certains auteurs ont d'ailleurs remarqué des différences dans les résultats et, à cette occasion, été sensibilisé à la propagation d'erreur d'arrondi. La norme IEEE 754 impose un codage particulier pour le résultat de certaines opérations arithmétiques qui peuvent générer une erreur. L'appellation *exception arithmétique* est alors utilisée pour des situations aussi diverses qu'un dépassement de capacité, une division par zéro, ou un calcul de racine carrée de nombre négatif.

L'évolution des ordinateurs, où fréquence du processeur et capacité de la mémoire augmentent régulièrement, permet d'accroître la taille des problèmes traités. Or la fréquence des accidents arithmétiques augmente elle aussi [68]. Si le domaine de variation de l'exposant et la taille de la mantisse restent constants, lorsque le nombre d'additions effectuées s'accroît, on constate une augmentation de la probabilité des dépassements de capacité et des cancellations [1]. Accroître la précision utilisée permet alors de diminuer cette probabilité, mais le n'annule pas. L'appellation *calcul impossible* [64] a même été attribuée à la résolution de certains problèmes sur ordinateur. C'est le cas du calcul de la suite (u_n) définie par [144] : $u_0 = 2$, $u_1 = -4$ et $u_{n+1} = 111 - 1130/u_n + 3000/(u_n u_{n-1})$. Bien que la limite exacte de la suite (u_n) soit 6, sur ordinateur, du fait des erreurs d'arrondi, cette suite semble converger très vite vers 100, quelle que soit la précision utilisée : si le calcul est effectué sur n chiffres, l'itéré u_{2n} vaut 100, à la précision du calcul près [146].

L'influence des erreurs dépend fortement de la stabilité du problème étudié. En l'occurrence, les erreurs d'arrondi peuvent avoir de graves conséquences dans les programmes modélisant les phénomènes chaotiques [152, 153, 154]. Elles peuvent parfois mener à une solution correspondant à d'autres conditions initiales que celles spécifiées dans le programme. Cet effet est parfaitement illustré par la suite (u_n) définie précédemment. La valeur 100 est une limite correspondant à d'autres conditions initiales u_0 et u_1. En effet, le polynôme $x^3 - 111x^2 + 1130x - 3000$ admet trois racines : 5, 6 et 100. Les erreurs d'arrondi font donc converger la suite vers une autre limite. Les trois limites possibles constituent néanmoins un ensemble stable. Les erreurs d'arrondi n'invalident pas nécessairement un programme. Un résultat non significatif, *i.e.* dont tous les chiffres sont entachés d'erreur d'arrondi, peut tout de même être révélateur de certaines propriétés mathématiques du problème [30]. D'ailleurs les équations de Navier-Stokes utilisées pour modéliser des phénomènes instables, par exemple en météorologie, peuvent générer des erreurs importantes, mais fournir des tendances justes.

1.2 Nos contributions

Nos axes de recherche sont essentiellement la fiabilité des logiciels numériques et le calcul à haute performance.

Nos travaux ont porté sur la propagation d'erreur d'arrondi lors de la résolution par différences finies de l'équation de la chaleur [89] et de l'équation des ondes [88]. Nous avons alors montré que l'évolution de l'erreur d'arrondi dans le temps est linéaire dans le cas de l'équation de la chaleur et au mieux linéaire dans le cas de l'équation des ondes. Ces travaux ont donné lieu à deux publications qui peuvent être consultées en 8.1 et en 8.2. Les résultats obtenus nous ont permis d'étudier la qualité numérique d'un code de propagation d'ondes acoustiques développé à l'Institut Français du Pétrole.

Les méthodes implicites de collocation ont été utilisées pour la résolution parallèle en temps d'équations différentielles [8, 9, 10, 12]. Nous avons étudié la résolution parallèle en temps et en espace des équations d'évolution par ces méthodes. En chaque point de l'espace, la solution d'une équation d'évolution est alors approchée par un polynôme dépendant du temps. Les

1. On appelle *cancellation* ou *élimination catastrophique* l'apparition brusque de chiffres entachés d'erreur d'arrondi lors de l'addition de deux nombres de signes contraires, voisins en valeur absolue.

différents polynômes, associés chacun à un point de l'espace discrétisé, peuvent être calculés en parallèle. De plus, pour un intervalle de temps fixé, la solution d'une équation d'évolution peut être approchée par ces polynômes en n'importe quel point de l'intervalle. Les différents polynômes peuvent donc être calculés en parallèle aux différents instants considérés. La résolution par méthode implicite de collocation de l'équation de la chaleur sur un domaine monodimensionnel a donné lieu à un article [90] qui peut être consulté en 8.3. Ces travaux ont été étendus par J. Kouatchou (Morgan State University, États-Unis) [100, 101, 102] à la résolution de l'équation de la chaleur par méthode implicite de collocation avec des schémas d'ordre élevé sur un domaine bidimensionnel.

Nos travaux ont en outre porté sur la qualité numérique de plusieurs codes développés à l'Institut Pierre Simon Laplace. La première étude consistait à analyser un code en Fortran simulant la circulation océanique dans un bassin. Ce code comportait environ 50 000 lignes réparties dans une centaine de fichiers. Il a fallu modifier ce code qui avait été développé pour calculateur CRAY, afin de pouvoir l'exécuter sur une station de travail. Puis d'autres modifications, assez fastidieuses, furent nécessaires pour pouvoir étudier la propagation d'erreur d'arrondi grâce à la bibliothèque CADNA [27]. Ont pu être analysées, d'une part l'évolution du nombre de chiffres significatifs exacts de diverses variables, d'autre part les instabilités numériques générées lors de l'exécution. La dernière phase de l'étude consistait à modifier les instructions responsables des principales instabilités. Cette étude n'a pas bénéficié des développements relatifs à la bibliothèque CADNA qui ont été effectués ultérieurement : la version vectorielle de CADNA pour calculateurs CRAY, l'outil modifiant automatiquement les codes en vue de leur exécution avec CADNA, le logiciel qui analyse automatiquement les instabilités numériques, le logiciel qui permet de visualiser la qualité numérique de résultats volumineux. Le rapport décrivant l'étude [93] peut être consulté en 8.4. Ces travaux se sont poursuivis par le stage ingénieur et le projet de fin d'études de F. Bonnet-Terras, au cours desquels a été étudiée, sur CRAY J90 et CRAY SV1, la qualité numérique du code dans sa configuration globale [23] et d'une version couplée avec un code de traceurs biologiques [24].

Comme de nombreux codes de simulation numérique sont exécutés sur machine vectorielle, nous avons porté la bibliothèque CADNA sur ce type de calculateur. La bibliothèque CADNA utilise un mode d'arrondi aléatoire : tous les résultats sont arrondis aléatoirement par excès ou par défaut. Il s'agissait donc que ce mode d'arrondi aléatoire n'inhibe pas la vectorisation. Deux versions vectorielles de CADNA ont été développées en Fortran : l'une pour NEC SX5, l'autre pour CRAY SV1. Le premier calculateur respecte la norme IEEE 754, mais pas le second. Des surcharges d'opérateurs ont été écrites pour les instructions utilisant la notation "tableau" du Fortran 90. De nombreux tests ont été effectués afin d'optimiser les performances de ces opérateurs vectoriels. Cependant, sur les deux calculateurs, nous avons été confrontés à des limitations du compilateur Fortran, notamment quant à ses possibilités d'*inlining*. Le portage de CADNA sur calculateur vectoriel a donné lieu à un article [94] qui peut être consulté en 8.5.

Le contrôle dynamique des méthodes d'approximation est un de nos principaux axes de recherche. Lorsque le pas de discrétisation utilisé dans une méthode numérique décroît, l'erreur de méthode diminue, mais l'erreur d'arrondi augmente généralement. Les stratégies que nous avons proposées permettent de calculer l'approximation pour laquelle l'erreur globale, qui comprend l'erreur de méthode et l'erreur d'arrondi, est minimale. De plus, nous montrons comment déterminer quels chiffres significatifs de l'approximation obtenue sont en commun avec le résultat exact. Nos premiers travaux dans cet axe de recherche [44], qui peuvent être consultés en 8.6, ont porté sur le contrôle dynamique de la méthode des trapèzes et de la méthode de Simpson. Nous y proposons une stratégie qui consiste à calculer une suite d'approximations jusqu'à ce que la différence entre deux itérés successifs soit non significative. L'article qui étend ces travaux au contrôle dynamique du calcul de suites à convergence linéaire ou exponentielle [91] peut être consulté en 8.7. Nous y montrons aussi comment combiner différents résultats théoriques afin de déterminer la validité d'une approximation nécessitant le calcul de plusieurs suites. Le contrôle dynamique de la méthode de Romberg [95] est décrit en 8.8 et celui de la méthode de Gauss-Legendre [34] en 8.9.

Le contrôle dynamique du calcul d'intégrales multiples a été étudié au cours de la thèse de M. Charikhi. Les stratégies que nous avons proposées utilisent deux approches. La première consiste, grâce au principe des "intégrales itérées", à décomposer une intégrale multiple en plusieurs intégrales monodimensionnelles qui peuvent être évaluées par une méthode de quadrature [92]. La seconde approche est fondée sur l'utilisation d'une formule de cubature, adaptée au domaine d'intégration.

La thèse de R. Adout traite de la convergence de la version redémarrée de la méthode de projection GMRES. Nous avons proposé une stratégie [4] qui permet d'accélérer cette convergence par des algorithmes d'extrapolation, tels le VEA (Vector Epsilon Algorithm). En outre, nous étudions le contrôle dynamique de la dimension des sous-espaces de Krylov sur lesquels s'effectuent les projections.

1.3 Organisation du document

Ce document décrit nos travaux les plus récents.

Dans le chapitre 2, nous présentons différentes méthodes d'analyse des erreurs d'arrondi, notamment l'approche probabiliste qui a conduit à la définition des arithmétiques stochastiques (l'une continue, l'autre discrète). Puis nous décrivons un résultat théorique essentiel pour le contrôle des opérations stochastiques.

Le chapitre 3 porte sur l'implantation de l'arithmétique stochastique discrète sur machine vectorielle. Nous présentons brièvement comment elle est implantée sur architecture séquentielle ou parallèle. Puis nous décrivons les tests effectués sur deux types de calculateur vectoriel afin d'optimiser les performances de arithmétique stochastique discrète en tenant compte de l'arithmétique du processeur (qui respecte ou non la norme IEEE 754) et des spécificités du compilateur.

De nombreuses méthodes numériques sont fondées sur le calcul d'une suite. Les stratégies décrites dans le chapitre 4 permettent alors de contrôler dynamiquement le nombre d'itérations effectuées. Nous obtenons alors une approximation pour laquelle l'erreur globale, formée de l'erreur de troncature et de l'erreur d'arrondi, est minimale. Nous montrons alors comment déterminer quels chiffres du résultat obtenu sont en commun avec la limite exacte.

Le chapitre 5 porte sur l'évaluation des intégrales multiples. La complexité du calcul croît alors avec la dimension de l'intégrale étudiée. Les stratégies proposées sont fondées sur un partitionnement du domaine, qui peut être contrôlé dynamiquement. Il est alors possible d'identifier les chiffres de l'approximation obtenue qui sont en commun avec la valeur exacte de l'intégrale.

En outre, nous présentons deux applications en physique qui ont pu être traitées grâce aux résultats décrits dans les chapitres 4 et 5. La première est le calcul d'une intégrale impropre établie en cristallographie, la seconde le calcul d'une intégrale multiple issue de la théorie des étoiles à neutrons.

Chapitre 2

Analyse d'erreur d'arrondi

Les résultats d'un programme exécuté en arithmétique à virgule flottante peuvent être invalidés du fait des erreurs d'arrondi. Celles-ci peuvent se propager progressivement : une accumulation régulière des erreurs d'arrondi est communément appelée *effet boule de neige*. Il peut aussi se produire une perte brutale de précision en une seule opération arithmétique : c'est le phénomène de *cancellation*. Même si d'autres arithmétiques existent [59], telles l'arithmétique multi-précision, les arithmétiques exactes ou les arithmétiques dynamiques, celles-ci restent plus coûteuses, en temps et en mémoire, et souvent plus difficiles à appréhender. L'arithmétique à virgule flottante demeurant très largement utilisée, plusieurs méthodes ont été développées, afin d'analyser la propagation d'erreur d'arrondi, voire même d'améliorer la qualité numérique des résultats. Après avoir introduit les différentes approches utilisées, nous présenterons les principes et les propriétés des arithmétiques stochastiques (l'une continue, l'autre discrète).

2.1 Les différentes approches

2.1.1 L'analyse directe

Pour effectuer une analyse des erreurs d'arrondi, on suppose que l'arithmétique à virgule flottante utilisée respecte la convention de l'*arrondi exact* [59], ce qui est le cas de tous les systèmes conformes à la norme IEEE 754 [86]. Grâce à la précision machine (*i.e.* l'erreur relative maximale due à un arrondi), l'erreur d'arrondi commise à l'issue d'un calcul peut être évaluée en tenant compte de l'erreur due à chaque opération. L'analyse directe permet d'estimer ou de majorer l'erreur directe, *i.e.* l'erreur finale due aux arrondis, en fonction des données et de la précision machine.

L'analyse directe a été mise en œuvre avec une approche statistique dans l'atelier AQUARELS [66, 67] via la méthodologie SCALP [73, 72].

J.H. Wilkinson a proposé dans [188] la *running error* analyse, fondée sur une approche déterministe, qui consiste à effectuer des majorations dynamiques de l'erreur directe [82, 189].

2.1.2 L'analyse inverse

L'analyse inverse ou régressive a été introduite par W. Givens [75] dans les années 1950, puis développée par J.H. Wilkinson [186, 187] dans les années 1960 pour analyser la fiabilité des calculs en précision finie. Ce domaine, décrit en détail dans [31, 82], a suscité de nombreuses études, notamment en algèbre linéaire numérique, et est toujours très actif.
L'analyse inverse permet de relier les calculs effectués sur ordinateur aux calculs exacts grâce au "principe de Wilkinson" selon lequel une solution calculée peut être considérée comme la solution exacte d'un problème proche. Ce principe offre plusieurs avantages. Il permet de :
- s'affranchir de l'arithmétique de l'ordinateur, puisque le résultat obtenu est la solution exacte d'un problème perturbé,
- mettre sur le même plan les erreurs d'arrondi et les erreurs sur les données initiales,
- distinguer dans la borne d'erreur obtenue la contribution de l'algorithme de celle du problème.

L'appellation "analyse inverse" provient du fait que les erreurs sont *réinjectées* dans le problème initial.

L'analyse inverse a été mise en œuvre dans le logiciel PRECISE [31, 32, 33, 156] avec une approche statistique qui consiste à perturber aléatoirement les données avec une amplitude croissante afin d'estimer l'erreur sur le résultat à partir d'un échantillon de calculs.
Les principes de l'analyse inverse sont appliqués avec une approche déterministe dans la bibliothèque LAPACK [13]. Cette bibliothèque, écrite en Fortran 77 permet la résolution de problèmes d'algèbre linéaire en fournissant des bornes d'erreur pour les valeurs qu'elle calcule.

2.1.3 Les méthodes fondées sur la différentiation algorithmique

Ces méthodes nécessitent le calcul des dérivées partielles de l'erreur directe globale par rapport aux erreurs d'arrondi élémentaires. Considérons l'évaluation en arithmétique à virgule flottante d'une fonction réelle f avec les données $X = (x_1,x_2,...,x_n)$. Supposons que le résultat $\hat{x}_N$ soit obtenu grâce au calcul des variables intermédiaires $\hat{x}_{n+1},...,\hat{x}_{N-1}$, ce qui génère les erreurs d'arrondi absolues élémentaires $\delta = (\delta_{n+1},...,\delta_N)$. L'erreur directe globale est :

$$\hat{x}_N - f(X) = \Delta_L - E_L, \tag{2.1}$$

où

$$\Delta_L = \sum_{k=n+1}^{N} \frac{\partial \hat{f}}{\partial \delta_k}(X,\delta)\, \delta_k. \tag{2.2}$$

Δ_L est l'approximation au premier ordre de l'erreur d'arrondi globale par rapport aux erreurs d'arrondi élémentaires δ_k et E_L est l'erreur de linéarisation associée à Δ_L.
En utilisant soit une borne [87, 127, 141] soit un modèle statistique [87, 126, 127, 172] pour les erreurs d'arrondi élémentaires, on peut obtenir une majoration ou une estimation, toutes deux du premier ordre, de l'erreur d'arrondi globale.

La méthode CENA (Correction des Erreurs Numériques d'Arrondi) [118, 119, 120, 122, 123], grâce au calcul sur ordinateur des erreurs d'arrondi élémentaires δ_k, fournit $\hat{\Delta}_L = fl(\Delta_L)$, l'approximation en arithmétique à virgule flottante de Δ_L. On en déduit le résultat corrigé $\overline{x}_N = fl(\hat{x}_N - \hat{\Delta}_L)$. Ce terme de correction linéaire est entaché d'erreur de troncature et d'erreur d'arrondi. Une borne de validité B_{E_C} du résultat corrigé est calculée dynamiquement par une *running error* analyse [82, 188, 189] de l'algorithme de différentiation et du processus de correction. On a alors $|\overline{x}_N - f(X)| \leq B_{E_C}$, en supposant que l'approximation au premier ordre de l'erreur globale soit valide. Cette hypothèse est satisfaite pour les algorithmes linéaires définis dans [118, 120, 123]. La méthode CENA a été testée sur de nombreux exemples linéaires ou non linéaires, notamment la sommation [123], le produit scalaire [120, 123] l'évaluation de polynômes [123], la résolution de systèmes linéaires [118, 120, 122, 123], l'approximation de racines de polynômes par l'itération de Newton [120, 124].

2.1.4 L'arithmétique d'intervalles

L'arithmétique d'intervalles est définie non pas sur des nombres réels, mais sur des intervalles. Les développements modernes de l'arithmétique d'intervalles ont commencé dans les années 1960 avec les travaux de R.E. Moore [160]. Sont apparus ensuite sur ce sujet des milliers d'articles de recherche et de nombreux livres parmi lesquels on peut citer [6, 108].

Lorsqu'elle est implantée en arithmétique à virgule flottante, l'arithmétique d'intervalles consiste à encadrer chaque nombre par un intervalle qui le contienne et qui puisse être représenté par des nombres machine[1]. Un tel intervalle est en général défini par deux nombres machine correspondant à ses bornes. Il peut l'être aussi par son centre et son rayon. Lorsque les opérandes d'une expression sont remplacées par des intervalles, le résultat calculé est un intervalle qui contient toutes les valeurs scalaires possibles pour toutes les valeurs scalaires contenues dans les opérandes. Les opérations arithmétiques et les fonctions mathématiques usuelles ont donc été étendues aux intervalles afin de garantir des bornes fiables pour l'intervalle résultant. Cette fiabilité est l'avantage majeur de l'arithmétique d'intervalles. Cependant l'intervalle obtenu est parfois trop large et dépend de la forme sous laquelle l'expression à calculer est écrite. La surestimation de l'erreur commise a plusieurs causes :

- l'accroissement régulier de la largeur des intervalles par rapport aux intervalles que l'on aurait obtenus en arithmétique exacte, qui est inhérent à l'implémentation de l'arithmétique d'intervalles en arithmétique à virgule flottante,
- le fait que deux variables corrélées (par exemple de même valeur) puissent être traitées comme étant décorrélées,
- l'effet enveloppant (ou "wrapping effect") qui survient lorsque l'image par une fonction d'un intervalle multidimensionnel (en dimension deux, un rectangle) n'est pas un intervalle multidimensionnel. L'intervalle multidimensionnel retenu doit alors inclure, "envelopper", cette image. Certaines techniques, tels les changements de variables, permettent de réduire l'effet enveloppant [128, 147].

Les algorithmes numériques utilisés en arithmétique d'intervalles diffèrent donc des algorithmes classiques.

1. Un nombre machine est un nombre réel représentable en machine de manière exacte.

S. Rump a montré [162] qu'en exprimant les intervalles sous la forme centre-rayon, l'arithmétique d'intervalles peut être entièrement implantée en utilisant les BLAS [2].

L'arithmétique d'intervalles est implantée dans plusieurs bibliothèques ou logiciels. Elle peut être mise en œuvre notamment grâce aux langages XSC. Le projet ACRITH, mené de 1980 à 1989 conjointement par l'équipe de U. Kulisch à l'université de Karlsruhe et la société IBM [85], a été à l'origine de l'écriture de ces langages. Il s'agit en fait de compilateurs ou de bibliothèques d'intervalles qui ont été développés à l'université de Karlsruhe et à l'université de Wuppertal à partir des langages Pascal, C ou C++ [79, 98, 99, 105]. Il est possible de télécharger sur internet [3] le compilateur Pascal-XSC et la bibliothèque C-XSC, mais aussi des bibliothèques où sont implantées différentes méthodes numériques en arithmétique d'intervalles pour l'évaluation de polynômes, la résolution de systèmes d'équations linéaires ou non linéaires, le calcul de coefficients de Taylor de fonctions analytiques, le calcul d'intégrales... Le Fortran-XSC est en cours de développement à l'université technique de Dresde.
Citons également la bibliothèque Intlib [96] pour les codes en Fortran, les modules Intpak [47] et intpakX [76, 104] pour l'environnement de calcul formel MAPLE, le module INTLAB [80, 163] pour le logiciel MATLAB.
L'arithmétique d'intervalles peut être utilisée en multiprécision, notamment grâce à la bibliothèque MPFI [161], la bibliothèque MPI_ARI [103] ou l'atelier AQUARELS [66, 67].

2.1.5 L'approche probabiliste

L'analyse directe et l'analyse inverse des erreurs d'arrondi, décrites auparavant, peuvent être mises en œuvre avec une approche statistique. Les méthodes d'estimation de l'erreur d'arrondi décrites dans cette partie consistent à perturber aléatoirement les résultats des opérations arithmétiques. Ces méthodes sont fondées sur l'arrondi aléatoire.

L'arrondi aléatoire

Dans les années 1950, G. Forsythe [69, 70, 71] a remarqué que les erreurs d'arrondi dans certains problèmes ne se comportent pas comme des variables aléatoires indépendantes et que de meilleures estimations de ces erreurs pourraient être obtenues si elles étaient effectivement des variables aléatoires indépendantes. Il propose un mode d'arrondi aléatoire qui consiste à arrondir par excès avec une probabilité v et par défaut avec la probabilité $1 - v$.
Ce mode d'arrondi aléatoire a été utilisé par T.E. Hull et J.R. Swenson [84] qui ont étudié la validité des modèles probabilistes de propagation d'erreurs d'arrondi. Il a également été utilisé par A.C. Callahan [29] pour des applications en traitement du signal.

En 1974, M. La Porte et J. Vignes [183] ont introduit la méthode des perturbations qui consistait alors à fixer le dernier bit de chaque résultat à 0 ou 1 aléatoirement. Puis une version légèrement modifiée de cette méthode a été implantée dans une fonction en Fortran

2. Les BLAS (Basic Linear Algebra Subroutines) sont des sous-programmes qui implantent des algorithmes d'algèbre linéaire tels le produit scalaire, la multiplication d'une matrice par un vecteur, la multiplication de deux matrices. Pour des raisons de performance, les BLAS ont une implémentation propre à chaque architecture. L'utilisation des BLAS dans un code assure sa portabilité.

3. `http://www.xsc.de`

nommée PER [176] qui consistait, avec une arithmétique en arrondi au plus près, à ajouter au dernier bit de chaque résultat 1 avec la probabilité 1/4, 0 avec la probabilité 1/2 et −1 avec la probabilité 1/4. Ces travaux ont permis à J. Vignes de développer la méthode CESTAC (Contrôle et Estimation Stochastique des Arrondis de Calculs) [178, 179] qui permet d'estimer le nombre de chiffres significatifs exacts de tout résultat calculé sur ordinateur. Cette méthode utilise un mode d'arrondi aléatoire défini ci-après.

Définition 1 *Chaque réel x qui n'est pas un nombre flottant est encadré par deux nombres flottants consécutifs X^- (valeur par défaut) et X^+ (valeur par excès). Le mode d'arrondi aléatoire définit comme nombre flottant X représentant x l'une des deux valeurs X^- ou X^+ avec la probabilité* 1/2.

Sur un processeur qui respecte la norme IEEE 754 [86] et permet l'utilisation de modes d'arrondi dynamiques, il est possible de simuler le mode d'arrondi aléatoire en sélectionnant aléatoirement le mode d'arrondi vers $+\infty$ ou $-\infty$ avant chaque opération arithmétique. Ce changement de mode d'arrondi peut être effectué simplement grâce à quelques lignes en assembleur.

Considérons maintenant le cas où le mode d'arrondi ne peut être modifié. Si celui-ci est l'arrondi vers zéro, il suffit d'ajouter 1, avec la probabilité 1/2, au dernier bit de la mantisse après chaque opération arithmétique, pour obtenir le même mode d'arrondi aléatoire que sur architecture IEEE. Si le mode d'arrondi est au plus près, afin que les erreurs d'arrondi soient centrées, une solution est d'ajouter +1 ou −1, avec la probabilité 1/2, au dernier bit de la mantisse du résultat de toute opération arithmétique. Un bit de précision est alors perdu pour le contrôle de l'erreur d'arrondi.

La méthode CESTAC

La méthode CESTAC repose sur une modélisation des résultats obtenus en arithmétique à virgule flottante. Il a été prouvé [38] qu'un résultat R calculé sur ordinateur peut être modélisé au premier ordre en 2^{-p} par :

$$R \approx Z = r + \sum_{i=1}^{n} g_i(d) 2^{-p} z_i. \tag{2.3}$$

r est le résultat exact, les $g_i(d)$ des constantes ne dépendant que des données et de l'algorithme, p le nombre de bits contenus dans la mantisse et les z_i des variables aléatoires indépendantes équiréparties sur $[-1,1]$.

Cette modélisation a deux conséquences importantes :

- l'espérance mathématique de la variable aléatoire Z est le résultat exact r,
- sous certaines hypothèses [42], la distribution de Z est quasi gaussienne (conséquence du théorème de la limite centrale).

Alors en identifiant R et Z, *i.e.* en négligeant les termes du second ordre, le test de Student peut être utilisé pour déterminer la qualité numérique de R. À partir de N échantillons R_i ($i = 1,2,...,N$), le test de Student fournit un intervalle de confiance sous une probabilité donnée pour l'espérance r.

$\forall \beta \in [0,1]$, $\exists \tau_\beta \in I\!R$ tel que

$$Pr\left(|\overline{R} - r| \leq \frac{\sigma \tau_\beta}{\sqrt{N}}\right) = 1 - \beta \tag{2.4}$$

où

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i \tag{2.5}$$

et

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2. \tag{2.6}$$

τ_β est le seuil de confiance de la distribution de Student pour $N-1$ degrés de liberté et une probabilité $1-\beta$.

Sous une probabilité $1-\beta$, le nombre de chiffres décimaux significatifs exacts de $\overline{R}$, *i.e.* le nombre de chiffres décimaux significatifs communs à $\overline{R}$ et à r, est minoré par

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}\,|\overline{R}|}{\sigma \tau_\beta}\right). \tag{2.7}$$

L'application de la méthode CESTAC à un programme qui fournit un résultat R consiste à :
- exécuter N fois le programme avec le mode d'arrondi aléatoire explicité dans la définition 1, on obtient alors N échantillons R_i de R,
- choisir la moyenne $\overline{R}$ de cette population comme résultat informatique final,
- estimer le nombre de chiffres significatifs exacts de $\overline{R}$ en utilisant l'équation (2.7).

En pratique, $N = 2$ ou $N = 3$ et $\beta = 0.05$. On a alors $\tau_\beta = 12.706$ pour $N = 2$ et $\tau_\beta = 4.4303$ pour $N = 3$.

Les équations (2.3) et (2.7) reposent sur deux principales hypothèses :
1. les erreurs d'arrondi α_i sont des variables aléatoires indépendantes équiréparties et centrées,
2. l'approximation au premier ordre en 2^{-p} est valide.

Concernant la première hypothèse, avec le mode d'arrondi aléatoire, les erreurs d'arrondi α_i sont des variables aléatoires. Cependant, en pratique elles ne sont pas rigoureusement centrées et dans ce cas le test de Student fournit une estimation biaisée du résultat calculé sur ordinateur. Il a été prouvé [45] qu'avec un biais de quelques σ l'erreur sur l'estimation du nombre de chiffres significatifs exacts de $\overline{R}$ est inférieure à un chiffre décimal. Donc même si la première hypothèse n'est pas rigoureusement satisfaite, l'estimation fournie par l'équation (2.7) reste fiable si on la considère exacte à un chiffre près.

Concernant la seconde hypothèse, dans la formulation des erreurs d'arrondi, les additions et les soustractions produisent uniquement des termes du premier ordre, mais les multiplications et les divisions génèrent aussi des termes du second ordre. Comme le modèle se limite au premier ordre, il est donc valable pour les additions et les soustractions. Pour qu'il reste valable pour les multiplications et les divisions, il faut que les erreurs du second ordre soient négligeables par rapport à celles du premier ordre. Il a été montré [38, 42] que si chaque opérande dans une multiplication ou le dénominateur dans une division est non significatif, alors l'approximation au premier ordre peut ne plus être valide. En pratique, la validation de la méthode CESTAC nécessite donc un contrôle dynamique des multiplications et des divisions pendant l'exécution du code. Ceci conduit à l'implémentation synchrone de la méthode, *i.e.* au calcul en parallèle des N échantillons R_i, et aussi au concept de zéro informatique [177].

Définition 2 *Lors de l'exécution d'un programme utilisant la méthode CESTAC, un résultat R, intermédiaire ou final, est un zéro informatique, noté* @.0 *si et seulement si l'une des deux conditions suivantes est vérifiée :*

1. $\forall i, R_i = 0$,
2. $C_{\overline{R}} \leq 0$.

Concrètement, soit R vaut zéro, soit R n'a aucun chiffre significatif exact (la valeur exacte correspondante est complètement masquée par le bruit numérique dû aux erreurs d'arrondi).

L'arithmétique stochastique a été définie à partir de l'implémentation synchrone de la méthode CESTAC et du concept de zéro informatique. En fait, on distingue l'arithmétique stochastique continue de l'arithmétique stochastique discrète. Toutes deux seront décrites en 2.2.

La bibliothèque CADNA [27], qui sera plus longuement décrite en 3.1, implante automatiquement l'arithmétique stochastique discrète dans les codes scientifiques écrits en Ada, C ou Fortran. Elle permet l'utilisation de nouveaux types numériques : les types stochastiques. À tout moment durant l'exécution, la qualité numérique des variables stochastiques peut être contrôlée. La bibliothèque CADNA permet de vérifier qu'aucune anomalie ne s'est produite pendant le calcul, en particulier dans les débranchements. En outre, l'utilisateur peut introduire une incertitude sur les données en réduisant leur précision initiale. La bibliothèque CADNA a permis, dans un contexte industriel ou académique, l'étude de la qualité numérique de programmes importants tels les codes fondés sur le modèle océanique OPA [135] comportant environ 50 000 lignes en Fortran [23, 24, 93].

L'arithmétique de Monte Carlo

L'arithmétique de Monte Carlo ou MCA (Monte Carlo Arithmetic) [149, 150, 151] consiste, comme la méthode CESTAC, à modéliser les résultats calculés sur ordinateur par des variables aléatoires. Elle offre plusieurs modes pour l'affectation et l'arrondi. Elle permet de transformer une valeur x en une valeur $\tilde{x}$ dite *inexacte* avec une précision maximale de t chiffres par

$$\tilde{x} = x + \beta^{e-t}\xi \tag{2.8}$$

où β représente la base et e l'exposant de x dans cette base.

ξ est une variable aléatoire pour laquelle plusieurs distributions sont proposées :

- identité
  la variable aléatoire ξ est fixée à 0,

- uniforme absolue
  la variable aléatoire ξ est distribuée uniformément sur $]-\frac{1}{2},\frac{1}{2}[$, d'espérance 0 et d'écart-type $\frac{1}{\sqrt{12}}$; $\tilde{x}$ est alors d'espérance x et d'écart-type $\frac{\beta^{e-t}}{\sqrt{12}}$,
- uniforme relative
  $\xi = \frac{x}{\beta^e}\,\delta$ où δ est une variable aléatoire distribuée uniformément sur $]-\frac{1}{2},\frac{1}{2}[$; $\tilde{x}$ est alors d'espérance x et d'écart-type $\frac{x\beta^{-t}}{\sqrt{12}}$,
- CESTAC
  il s'agit en fait de la méthode des perturbations telle qu'elle avait été implantée dans la fonction en Fortran PER [176].

La transformation spécifiée par l'équation (2.8) peut être appliquée aux opérandes et au résultat d'une opération arithmétique. Plusieurs modes d'arrondi sont proposés pour le résultat obtenu :

- l'arrondi au plus près,
- l'arrondi par excès ou par défaut avec la probabilité $\frac{1}{2}$,
- l'arrondi par excès ou par défaut avec une probabilité proportionnelle à la distance qui sépare le résultat de chacune des deux valeurs possibles.

Comme avec la méthode CESTAC, N exécutions d'un programme avec MCA fournissent N résultats différents R_i $(i = 1,2,...,N)$. Il s'agit de N échantillons d'une distribution dont l'espérance E et la variance peuvent être estimées par $\overline{R}$ et σ^2 conformément aux équations (2.5) et (2.6). L'erreur dite *standarde* sur $\overline{R}$ peut alors être estimée par $\frac{\sigma}{\sqrt{N}}$.
Comme $\overline{R}$ est la moyenne des variables aléatoires indépendantes R_i, pour les valeurs de N pas trop élevées, d'après le théorème de la limite centrale, la distribution de $\overline{R}$ est gaussienne. On peut alors, comme avec la méthode CESTAC, utiliser le test de Student pour déterminer un intervalle de confiance sous une probabilité donnée.

$$Pr\left(|\overline{R} - E| \leq \frac{\sigma\tau_\beta}{\sqrt{N}}\right) = 1 - \beta \tag{2.9}$$

où τ_β est le seuil de confiance de la distribution de Student pour $N-1$ degrés de liberté et une probabilité $1-\beta$.

Un logiciel implantant en langage C l'arithmétique de Monte Carlo peut être téléchargé sur internet[4]. Son auteur, D.S. Parker, a voulu que ce logiciel soit ouvert. L'utilisateur peut affecter la précision de son choix à chaque opérande et au résultat d'une opération arithmétique. Ceux-ci peuvent de plus être perturbés selon l'une des quatre techniques spécifiées auparavant. Le résultat peut alors être arrondi selon trois modes au choix. Le nombre N de résultats fournis par MCA est lui aussi choisi par l'utilisateur. Le comportement d'un code peut donc différer d'une exécution à l'autre du fait des débranchements. L'utilisateur doit alors tirer lui-même les conclusions de ces différences. Contrairement à CADNA, le logiciel conçu par D.S. Parker, ne permet pas de surcharge d'opérateurs. L'utilisateur doit réécrire en utilisant des fonctions spécifiques toutes les opérations arithmétiques. Coûteux en temps d'exécution,

4. `http://www.cs.ucla.edu/~stott/mca/MCAdemo.c`

ce logiciel est clairement voué à des études expérimentales, menées jusqu'à présent sur des problèmes simples. Ceci est d'ailleurs confirmé par D.S. Parker dans [149] :

> MCA is aimed at encouraging the experimental outlook of Monte Carlo computation in numerical computing, and at extending standard hardware floating-point implementations to support this outlook. It makes a point of avoiding statistical interpretation of its results, leaving this interpretation to the user, or to statistical analysis software.

Dans le même document, D.S. Parker reconnaît la simplicité de CADNA qui, grâce à la surcharge d'opérateurs, nécessite peu de modifications des codes à analyser. Un autre atout de CADNA réside dans l'implémentation des relations d'ordre de l'arithmétique stochastique discrète (qui seront spécifiées en 2.2.2). Celles-ci permettent la prise en compte des instabilités qui surviennent lors des débranchements. J. Vignes [181] a effectué avec CADNA les expérimentations numériques décrites dans [149, 151] et a obtenu des résultats identiques à ceux fournis par MCA.

2.2 Principe des arithmétiques stochastiques

L'arithmétique stochastique a été définie à partir de l'implémentation synchrone de la méthode CESTAC. En fait, il existe deux types d'arithmétique stochastique : elle peut être soit continue, soit discrète. L'arithmétique stochastique continue s'applique à des nombres définis par l'espérance et la variance d'une variable aléatoire gaussienne, tandis que l'arithmétique stochastique discrète s'applique à des N-uplets fournis par la méthode CESTAC.

2.2.1 L'arithmétique stochastique continue

L'arithmétique stochastique continue est une modélisation de l'implémentation synchrone de la méthode CESTAC. Avec cette implémentation, les N exécutions d'un code sont effectuées en parallèle. Les N résultats de chaque opération arithmétique peuvent alors être considérés comme des réalisations d'une variable aléatoire gaussienne centrée sur le résultat exact. On peut alors définir un nouveau nombre, appelé *nombre stochastique* et une nouvelle arithmétique, appelée *arithmétique stochastique (continue)* qui s'applique à ces nombres. On peut aussi définir une nouvelle égalité et de nouvelles relations d'ordre, qui prennent en compte le nombre de chiffres significatifs exacts des opérandes stochastiques.

Un nombre stochastique X est noté (m,σ^2), où m est l'espérance de X et σ son écart-type. Les opérations arithmétiques stochastiques ($s+$, $s-$, $s\times$, $s/$) correspondent aux termes du premier ordre en $\frac{\sigma}{m}$ des opérations entre deux variables aléatoires gaussiennes indépendantes.

Définition 3 *Soient* $X_1 = (m_1,\sigma_1^2)$ *et* $X_2 = (m_2,\sigma_2^2)$*. Les opérations arithmétiques stochas-*

tiques sur X_1 *et* X_2 *sont définies par :*

$$X_1 \; s{+} \; X_2 = \left(m_1 + m_2 \; , \; \sigma_1^2 + \sigma_2^2\right) \tag{2.10}$$
$$X_1 \; s{-} \; X_2 = \left(m_1 - m_2 \; , \; \sigma_1^2 + \sigma_2^2\right) \tag{2.11}$$
$$X_1 \; s{\times} \; X_2 = \left(m_1 \times m_2 \; , \; m_2^2\sigma_1^2 + m_1^2\sigma_2^2\right) \tag{2.12}$$
$$X_1 \; s/ \; X_2 = \left(m_1/m_2 \; , \; \left(\frac{\sigma_1}{m_2}\right)^2 + \left(\frac{m_1\sigma_2}{m_2^2}\right)^2\right) \; \text{avec } m_2 \neq 0. \tag{2.13}$$

On peut associer à chaque nombre stochastique une précision. Si $X = (m,\sigma^2)$, il existe λ_β (dépendant seulement de β) tel que

$$P\left(X \in [m - \lambda_\beta\sigma \; , \; m + \lambda_\beta\sigma]\right) = 1 - \beta, \tag{2.14}$$

$I_{\beta,X} = [m - \lambda_\beta\sigma \; , \; m + \lambda_\beta\sigma]$ est l'intervalle de confiance de m à $1 - \beta$. Le nombre de chiffres significatifs décimaux communs à tous les éléments de $I_{\beta,X}$ et à m est minoré par :

$$C_{\beta,X} = \log_{10}\left(\frac{|m|}{\lambda_\beta\sigma}\right). \tag{2.15}$$

La définition suivante est la modélisation du concept de zéro informatique, introduit précédemment.

Définition 4 *Un nombre stochastique* X *est un zéro stochastique, noté* $\underline{0}$*, si et seulement si*

$$C_{\beta,X} \leq 0 \quad ou \quad X = (0,0).$$

À partir du concept de zéro stochastique, un nouveau concept d'égalité et de nouvelles relations d'ordre ont été définis.

Définition 5 *Soient* $X_1 = (m_1,\sigma_1^2)$ *et* $X_2 = (m_2,\sigma_2^2)$ *deux nombres stochastiques.*

- *L'égalité stochastique, notée* $s{=}$*, est définie par :*
  $X_1 \; s{=} \; X_2$ *si et seulement si* $X_1 \; s{-} \; X_2 = \underline{0}$.
- *Les inégalités stochastiques, notées* $s{>}$ *et* $s{\geq}$ *sont définies par :*
  $X_1 \; s{>} \; X_2$ *si et seulement si* $m_1 > m_2$ *et* $X_1 \; s{\neq} \; X_2$,
  $X_1 \; s{\geq} \; X_2$ *si et seulement si* $m_1 \geq m_2$ *ou* $X_1 \; s{=} \; X_2$.

L'arithmétique stochastique continue est une modélisation de l'arithmétique de l'ordinateur qui prend en compte les erreurs d'arrondi. Les propriétés de l'arithmétique stochastique continue [41, 42] ont mis en évidence les différences théoriques entre l'arithmétique approximative de l'ordinateur et l'arithmétique exacte.

2.2.2 L'arithmétique stochastique discrète ou ASD

En Arithmétique Stochastique Discrète (ASD), un nombre réel devient un N-uplet et une opération entre N-uplets se fait élément par élément en utilisant à chaque fois le mode d'arrondi aléatoire. Le nombre de chiffres significatifs exacts d'un tel N-uplet peut être estimé grâce à l'équation (2.7). À partir du concept de zéro informatique introduit précédemment, un concept d'égalité et des relations d'ordre ont été définis pour l'ASD.

Définition 6 *Soient X et Y deux N-uplets fournis par la méthode CESTAC.*

- *L'égalité stochastique discrète, notée $ds=$, est définie par :*
  $X ds= Y$ *si et seulement si* $X - Y = @.0$.
- *Les inégalités stochastiques discrètes, notées $ds>$ et $ds\geq$ sont définies par :*
  $X ds> Y$ *si et seulement si* $\overline{X} > \overline{Y}$ *et* $X ds\neq Y$,
  $X ds\geq Y$ *si et seulement si* $\overline{X} \geq \overline{Y}$ *ou* $X ds= Y$.

Les relations d'ordre en ASD sont essentielles pour le contrôle des débranchements. Du fait des erreurs d'arrondi, si A et B sont deux résultats informatiques issus de séquences de calcul et a et b les valeurs exactes correspondantes,

$$a > b \not\Rightarrow A > B \quad \text{et} \quad A > B \not\Rightarrow a > b.$$

Cette complète décorrélation est à la base de nombreux problèmes de la programmation scientifique : test d'arrêt non vérifié, bouclage de programme en géométrie algorithmique, etc. La prise en compte de la qualité numérique des opérandes dans les relations d'ordre permet de résoudre en partie ces difficultés [41].
L'ASD permet donc d'estimer l'impact des erreurs d'arrondi dans tout résultat de programmes scientifiques et de contrôler le bon déroulement d'un programme, notamment dans le cas de débranchements instables. L'ASD est implantée dans la bibliothèque CADNA [27].

On peut mettre en relation la précision d'un nombre stochastique et le nombre de chiffres significatifs exacts d'un N-uplet fourni par la méthode CESTAC. En effet, lorsque la valeur de N est faible (2 ou 3), ce qui est le cas en pratique, les valeurs fournies par les équations (2.7) et (2.15) sont très proches. Elles représentent dans un résultat calculé sur ordinateur le nombre de chiffres significatifs qui ne sont pas entachés d'erreurs d'arrondi. Les deux types d'arithmétiques stochastiques sont donc cohérentes. Les propriétés établies dans le cadre théorique de l'arithmétique stochastique continue peuvent s'appliquer sur ordinateur via l'utilisation de l'ASD.

2.3 Contrôle des opérations stochastiques

Le théorème 1 a été établi en arithmétique stochastique continue. Il permet de comparer les résultats d'opérations arithmétiques stochastiques avec ceux obtenus lorsque les opérations arithmétiques classiques sont effectuées avec les valeurs exactes correspondantes.

Considérons une méthode numérique dont le but est l'approximation d'une valeur exacte x. Cette méthode peut, par exemple, consister à calculer l'itéré d'une suite qui converge vers x. Même en utilisant une arithmétique à précision infinie, la valeur obtenue n'est pas x, mais une approximation entachée d'erreur de troncature. Lorsqu'une telle méthode numérique est utilisée, nous comparons l'approximation obtenue en arithmétique stochastique avec la valeur exacte correspondante.

Théorème 1 *Soit $X_1 = (m_1,\sigma_1^2)$ l'approximation d'une valeur exacte x_1 en arithmétique stochastique continue. Supposons que les bits significatifs exacts de X_1, i.e. ceux qui ne sont pas entachés d'erreur d'arrondi, sont en commun avec x_1, à p près.*
De même, soit $X_2 = (m_2,\sigma_2^2)$ l'approximation en arithmétique stochastique continue d'une valeur exacte x_2, dont les bits significatifs exacts sont en commun avec x_2, à q près.
Soit $\bigcirc$ un opérateur arithmétique : $\bigcirc \in \{+, - , \times ,/\}$ et soit $s\bigcirc$ l'opérateur stochastique correspondant : $s\bigcirc \in \{s+ , s- , s\times , s/\}$.
Alors les bits significatifs exacts de $X_1 \, s\bigcirc \, X_2$ sont en commun avec la valeur exacte $x_1 \bigcirc x_2$, à max(p,q) près.

La preuve de ce théorème, établie dans [91], peut être consultée en 8.7.

Le théorème 1 permet de contrôler les opérations arithmétiques effectuées à partir de résultats de méthodes numériques obtenus sur ordinateur. Ce théorème a été prouvé pour les opérations arithmétiques stochastiques, qui sont une modélisation des opérations effectuées dans l'implémentation synchrone de la méthode CESTAC. En pratique, le théorème 1 est utilisé pour les résultats obtenus en ASD. Des applications du théorème 1 seront décrites notamment dans le chapitre 4 dans le cadre du calcul de suites en ASD.

2.4 Conclusion et perspectives

Au cours de ces dernières années, bien que divers travaux aient porté sur les méthodes probabilistes d'analyse d'erreur d'arrondi, celles-ci pourront faire l'objet de nouvelles études théoriques.

En étudiant l'addition et la multiplication par un scalaire en arithmétique stochastique continue, S. Markov et R. Alt ont récemment mis en évidence de nouvelles propriétés des nombres stochastiques [139].
Le théorème 1, donné en 2.3, permet, lors d'une opération arithmétique, de déduire des chiffres significatifs du résultat qui ne sont pas entachés d'erreur d'arrondi ceux qui sont en commun avec le résultat exact. Établi en arithmétique stochastique continue, il est utilisé en pratique pour des résultats calculés en ASD.
À partir du concept de zéro informatique, décrit en 2.1.5, a été proposé un test d'arrêt optimal pour les algorithmes itératifs [7, 178, 179, 180, 182]. Dans les chapitres 4 et 5, nous montrons que la combinaison du test d'arrêt optimal et de nouveaux résultats théoriques, tels le théorème 1, permet de prendre en compte à la fois l'erreur d'arrondi et l'erreur de troncature dans différentes méthodes numériques.
En arithmétiques d'intervalles, les opérateurs classiques, appelés "opérateurs externes", peuvent engendrer un accroissement trop important de la largeur des intervalles résultats. S. Markov a introduit les opérateurs internes [166, 137, 138]. Selon la monotonicité des fonctions calculées, on peut choisir l'un ou l'autre type d'opérateur, afin de réduire l'accroissement de la largeur des intervalles. R. Alt et J.-L. Lamotte ont proposé une arithmétique d'intervalles aléatoire qui consiste à choisir aléatoirement l'opérateur interne ou externe. Les résultats expérimentaux sur la distribution des intervalles ainsi obtenus [11] devront être validés par une étude théorique.

En tenant compte du comportement particulier des erreurs d'arrondi en arithmétique à virgule fixe, utilisée notamment sur les calculateurs embarqués, de nouvelles propriétés de l'arithmétique stochastique pourraient être établies pour ce type de calculateur.
Une autre perspective est la prise en compte de l'erreur d'arrondi générée par les fonctions élémentaires qui ne fournissent pas toujours l'arrondi exact du résultat [62, 125, 145].

Chapitre 3

Implémentation de l'ASD sur architecture vectorielle

Apparues dans les années 1970, les machines vectorielles sont toujours très utilisées pour la simulation numérique. Les résultats d'une enquête réalisée par l'IDRIS [1] en 2004 auprès de ses utilisateurs ont montré leur attachement à ce type de calculateur. Ce chapitre porte sur l'implémentation de l'Arithmétique Stochastique Discrète (ASD) sur machine vectorielle. Nous présentons le fonctionnement du logiciel CADNA [27], qui implante l'ASD, sur architectures séquentielle et parallèle. Puis nous décrivons le portage de ce logiciel sur deux types de calculateur vectoriel : CRAY SV1 et NEC SX5. Nous présentons notamment les efforts accomplis pour améliorer les performances de CADNA sur ces deux machines.

3.1 Le logiciel CADNA

3.1.1 Sur architecture séquentielle

L'arithmétique stochastique discrète est implantée dans le logiciel CADNA, dont le premier but est l'estimation des erreurs d'arrondi dans tout résultat de programmes scientifiques. Le logiciel CADNA permet de connaître à tout moment le nombre de chiffres significatifs exacts de tout résultat intermédiaire.
En utilisant les définitions des relations d'ordre et d'égalité de l'arithmétique stochastique discrète, le logiciel CADNA permet aussi de contrôler les débranchements, ce qui constitue le deuxième but du logiciel.
Le troisième but est de permettre un véritable débogage numérique. CADNA détecte en permanence l'apparition d'instabilités numériques et génère dans ce cas une trace dans un fichier auxiliaire. L'utilisation du débogueur symbolique permet, dès lors, de retrouver dans le fichier source l'instruction responsable de l'instabilité.
De plus, afin de garantir la fiabilité de l'estimation des erreurs d'arrondi, CADNA peut intégrer une auto-validation, c'est-à-dire une vérification permanente des hypothèses de va-

1. Institut du Développement et des Ressources en Informatique Scientifique, Bâtiment 506, BP 167, 91403 Orsay cedex, France.

lidité de la méthode CESTAC. Le logiciel est capable de déterminer à quels moments ces hypothèses ne sont plus vérifiées et, si tel est le cas, d'en avertir l'utilisateur.

J.-M. Chesneaux a réalisé la première version du logiciel CADNA en langage ADA [36, 37, 39], puis il a écrit une version en Fortran 90 [40]. Celle-ci s'appliquait initialement à des programmes écrits en Fortran 77. Puis une version prenant en compte les notations "tableau" du Fortran 90 a été développée à l'occasion du portage de CADNA sur architecture vectorielle (décrit en 3.2). Une version en langage C++ a en outre été écrite par J.-L. Lamotte. Une version en Fortran et une version en C++ du logiciel CADNA sont disponibles sur internet[2].

CADNA se présente comme une bibliothèque utilisable après la compilation lors de l'édition de liens. Elle permet au programmeur de disposer de trois nouveaux types numériques - les types stochastiques, sur lesquels s'effectue le contrôle des erreurs d'arrondi. Ces types sont :
- type (SINGLE_ST) : stochastique simple précision
- type (DOUBLE_ST) : stochastique double précision
- type (COMPLEX_ST) : complexe stochastique simple précision.

CADNA contient les définitions de tous les opérateurs arithmétiques, de relation ainsi que les fonctions élémentaires concernant les types stochastiques. Grâce à la surcharge d'opérateurs, l'utilisation de CADNA sur d'anciens programmes nécessite très peu de modifications. Dans une expression arithmétique, on peut mélanger les types stochastiques, les types flottants, et les types entiers. Les règles de prédominance s'appliquent avec en plus la prédominance des types stochastiques sur les autres. En sortie, seuls les chiffres significatifs exacts des variables stochastiques sont affichés ce qui permet une visualisation très simple de la précision.

L'implémentation de CADNA sur une architecture séquentielle consiste à "plonger" le calcul dans $\mathbb{R}^N$, où la valeur de N dépend des versions du logiciel. Pour les versions académiques disponibles sur internet, $N = 3$ et pour les versions commercialisées qui sont plus optimisées et destinées à être utilisées dans un contexte industriel, $N = 2$. Cette solution simple permet de simuler parfaitement l'implémentation synchrone de la méthode CESTAC en évitant tout problème de synchronisation ou de rendez-vous. Chaque opération est effectuée N fois avant de passer à la suivante. Ainsi, à tout instant, on dispose des N représentants de chaque variable stochastique ce qui est indispensable pour l'estimation de la précision de tout résultat intermédiaire, pour la validation de la méthode CESTAC et pour le contrôle des débranchements.

L'utilisation de CADNA a un coût en temps d'exécution qui dépend du code à valider, de la valeur du paramètre N, du choix d'activer ou non l'auto-validation. Avec une version de CADNA sans auto-validation où $N = 2$, ce coût est d'un facteur compris entre 3 et 6 sur architecture séquentielle.

Une chaîne de validation numérique qui intègre le logiciel CADNA a été développée lors de ces dernières années. Un atelier de validation numérique convivial et puissant devait présenter les caractéristiques suivantes :

1. une implémentation automatique de l'ASD dans les codes sources de l'utilisateur,
2. une exécution qui tienne compte de la machine utilisée, quelle que soit son architecture,

2. `http://www.lip6.fr/cadna`

3. une analyse automatique de l'information fournie par le logiciel.

Le développement d'un outil réalisant le premier point provient du constat suivant. Les modifications à apporter à un code en vue de sa validation par le logiciel CADNA sont peu nombreuses : elles concernent essentiellement les déclarations de variables et les entrées-sorties. Néanmoins ces modifications peuvent s'avérer fastidieuses si les codes sources sont importants et comportent de nombreux sous-programmes. Un outil, développé par la société ADALOG, instrumente automatiquement les codes sources en C ou en Fortran pour leur exécution avec le logiciel CADNA.

Le troisième point a suscité le développement de deux types d'outils.
Le logiciel CADNA fournit dans un fichier la liste des instabilités numériques survenues lors d'une exécution. Si ce fichier est volumineux, son analyse peut s'avérer problématique. Le logiciel CADTRACE, développé par J.-L. Lamotte, fournit le nombre d'instabilités de chaque type (division instable, multiplication instable, perte brutale de précision, ...) et l'instruction qui a généré chacune de ces anomalies.
Le logiciel CADNA permet d'afficher le nombre de chiffres significatifs exacts des résultats. Cet affichage peut devenir confus si le résultat attendu est un tableau volumineux. Un prototype de logiciel de visualisation, nommé CADVISU, a donc été créé [111] afin de traiter les tableaux bidimensionnels. Il attribue à chaque élément du tableau une couleur correspondant à son nombre de chiffres significatifs exacts.

Nous développerons ci-après le second point. Nous décrirons le portage du logiciel CADNA sur architecture parallèle, puis sur architecture vectorielle.

3.1.2 Sur architecture parallèle

Deux types de versions du logiciel CADNA ont été développées sur architecture parallèle. On distingue celles qui permettent de contrôler la qualité numérique des codes parallèles de celles qui mettent à profit les différents processeurs du calculateur pour accroître les performances de CADNA sur les codes séquentiels.

Validation numérique de codes parallèles

En 1994 et 1995, une version de CADNA a été développée [15, 110] pour la validation numérique des codes parallèles en Fortran 77 qui utilisent la bibliothèque d'échange de messages PVM (Parallel Virtual Machine). Cette implémentation a posé deux problèmes :

- comme CADNA utilise les spécificités du Fortran 90 concernant les types dérivés, il a fallu redéfinir une nouvelle interface adaptée au Fortran 90 pour les fonctions de PVM version 3 ;
- comme PVM version 3 n'autorisait pas la définition de nouveaux types, l'ajout des types stochastiques dans le noyau de PVM a été nécessaire.

Cette version de CADNA a permis de contrôler la qualité numérique d'un modèle de diffraction en rayons X [114]. Le code étudié traitait des données d'une telle taille (de 1 à 7 Go) que seule une version parallèle avait été développée.

Une version de CADNA permettant la validation numérique des codes parallèles qui utilisent MPI (Message Passing Interface) a été écrite par la suite. Comme MPI offre la possibilité de créer de nouveaux types, le développement de cette version a été plus aisé.

Ces implémentations sont toutes deux fondées sur l'ajout d'une interface pour les types stochastiques, afin d'intégrer les communications dans le système de validation. Chaque processus du code parallèle est alors assimilé à un programme séquentiel.

Parallélisation de l'ASD

Plusieurs versions parallèles du logiciel CADNA ont été développées afin d'en réduire le coût en temps d'exécution. Dans toutes les implémentations effectuées, une variable stochastique correspond à trois flottants. Le principe est l'exécution en parallèle des trois processus qui effectuent chacun le calcul d'un des éléments formant une variable stochastique. Un système de communication permet alors à ces processus d'échanger des informations. Les machines cibles doivent donc présenter de bonnes performances pour l'échange de petites données entre les différents processus de calcul.

Un premier prototype nommé RAPP (Random Arithmetic Parallel Prototype) a été réalisé à partir de 1993 par A. Asserrhine au cours de sa thèse [14, 15]. Ce prototype a été implanté en langage Occam[3] sur une machine composée de Transputers T800 et en Fortran 90 sur Connection Machine CM5.

D'autres implémentations [112] ont ensuite été effectuées en C++, en utilisant la bibliothèque MPI ou le système IPC[4], sur des machines :

- à mémoire partagée (machines quadri-processeurs SUN ou HP)
- à mémoire distribuée (la machine Multi-PC[5] du LIP6 fondée sur un réseau Fast-HSL et la machine BIP [158] de l'ENS-Lyon qui utilise un réseau Myrinet).

Une interface identique à celle de la version séquentielle a alors été développée afin de masquer complètement le parallélisme pour l'utilisateur. Pour valider un code séquentiel, la version séquentielle ou la version parallèle de CADNA peut être utilisée, le choix se faisant à la compilation et à l'édition de liens.
Ces implémentations souffrent d'un défaut : elles n'activent pas l'auto-validation de la méthode CESTAC, car dans ce cas les performances deviennent catastrophiques. Elles ont néanmoins l'avantage de fixer un objectif de performance pour l'implémentation de la méthode CESTAC avec auto-validation.

De nouveaux tests ont ensuite été effectués [115] sur le calculateur IBM Regatta Power 4 de l'IDRIS. Des processus d'auto-validation ont été ajoutés aux programmes précédemment écrits avec MPI. Ceux-ci se distinguent des trois processus de calcul qui implantent l'arithmétique aléatoire synchrone. Les résultats intermédiaires nécessaires à l'auto-validation de la méthode CESTAC sont envoyés aux processus de validation qui ont un fonctionnement

3. Il s'agit du nom d'un membre de l'ordre franciscain au 13ème siècle ! Celui-ci fut adopté dans les années 1980 par la société Inmos Ltd pour identifier son nouveau langage de programmation pour les Transputers.

4. InterProcess Communication, permet le partage de zones mémoire entre différents processus sous UNIX système V.

5. `http://mpc.lip6.fr`

indépendant. Il apparaît que les temps de calcul obtenus avec la version parallèle et auto-validée de CADNA sont quasiment divisés par deux dans le meilleur des cas par rapport à ceux obtenus avec la version séquentielle. Dans la pratique, seulement deux processus supplémentaires sont nécessaires pour l'auto-validation de la méthode CESTAC. Cinq processeurs suffisent pour valider de manière efficace un programme.

3.2 Portage de CADNA sur architecture vectorielle

Le logiciel CADNA a été porté sur deux types de calculateurs vectoriels : NEC SX5 et CRAY SV1. Le premier respecte la norme IEEE 754, mais pas le second. L'ASD y a été implantée en Fortran. Les types stochastiques sont donc des types dérivés. Dans les deux implémentations effectuées, chaque variable A de type stochastique est une structure contenant deux flottants notés $A\%x$ et $A\%y$.

Soient A et B deux nombres stochastiques :

$$A = (A\%x, A\%y) \;\text{ et }\; B = (B\%x, B\%y). \tag{3.1}$$

Soit ω l'une des quatre opérations arithmétiques $(+, - , * ,/)$ effectuée en arrondi vers $+\infty$ ou $-\infty$ avec la probabilité $1/2$. On désigne par Ω l'opération correspondante sur les nombres stochastiques :

$$A\;\Omega\;B = (A\%x\;\omega\;B\%x, A\%y\;\omega\;B\%y). \tag{3.2}$$

Les opérations $A\%x\;\omega\;B\%x$ et $A\%y\;\omega\;B\%y$ sont effectuées avec des modes d'arrondi opposés, ce qui est recommandé pour augmenter l'efficacité de la méthode CESTAC.

Nous décrivons ci-après les spécificités de chacune des deux implémentations.

3.2.1 Sur architecture vectorielle non-IEEE

L'ASD a été implantée sur le calculateur vectoriel CRAY SV1 du CCR [6]. Celui-ci ne respecte pas la norme IEEE 754. Tous les flottants y sont codés sur 64 bits : 1 pour le signe, 15 pour l'exposant et 48 pour la mantisse. L'addition et la soustraction sont effectuées en arrondi vers zéro, la multiplication et la division en arrondi au plus près (plus ou moins...).

L'ASD, sur CRAY SV1, nécessite de perturber la mantisse de chaque résultat d'une manière qui dépend de l'opération arithmétique effectuée. Le résultat d'une addition ou d'une soustraction est perturbé en ajoutant 0 ou 1 au dernier bit de la mantisse. Le résultat d'une multiplication ou d'une division est perturbé en ajoutant -1 ou 1 au dernier bit de la mantisse.

Après une opération vectorielle, les éléments du tableau résultat ne doivent pas tous être perturbés de la même manière. La perturbation de leurs mantisses doit être effectuée aléatoirement.

Considérons l'opération vectorielle stochastique $C = A\Omega B$. L'opérateur vectoriel Ω est surchargé, comme explicité dans la relation (3.2). Les tableaux $A\%x\;\omega\;B\%x$ et $A\%y\;\omega\;B\%y$

6. Centre de Calcul et de Recherche, 4 place Jussieu, 75252 Paris cedex 05, France.

sont perturbés au moyen d'opérations logiques vectorielles. Si Ω représente une addition ou une soustraction stochastique, quatre opérations flottantes et quatre opérations logiques sont effectuées. Si Ω représente une multiplication ou une division stochastique, quatre opérations flottantes et huit opérations logiques sont alors nécessaires.

Le tableau 3.1 présente les temps d'exécution sur CRAY SV1 d'un code effectuant la multiplication de deux matrices de dimension 1 000. Les performances obtenues avec CADNA sont exprimées en Mstops : millions d'opérations stochastiques par seconde. Les résultats correspondant à la version scalaire de CADNA ont été obtenus avec un code utilisant des boucles "DO". Les résultats correspondant à la version vectorielle de CADNA ont été obtenus avec un code utilisant les notations "tableau" propres à Fortran 90.

Le tableau 3.2 présente les temps d'exécution sur CRAY SV1 d'un code effectuant la décomposition LU sans pivotage d'une matrice de dimension 1000.

Dans les deux exemples, on remarque un facteur d'environ 8 entre les performances de la version scalaire de CADNA et celles de la version vectorielle. Bien que la bibliothèque CADNA soit *inlinée*, une version du code, où tous les opérateurs vectoriels stochastiques sont *inlinés* manuellement, a été développée. On remarque un facteur 1.8 entre les performances de cette version et celles où la version vectorielle de CADNA est utilisée, dû à de multiples recopies de tableaux.

	Temps d'exécution	Performances
code vectoriel classique	4.3 s	465 Mflops
CADNA, version scalaire	1 088 s	1.8 Mstops
CADNA, version vectorielle	124 s	16 Mstops
CADNA, avec inlining manuel	68 s	29 Mstops

TAB. 3.1 – *Performances d'un code effectuant une multiplication de matrices*

	Temps d'exécution	Performances
code vectoriel classique	7.1 s	93.9 Mflops
CADNA, version scalaire	395 s	1.7 Mstops
CADNA, version vectorielle	49 s	14 Mstops
CADNA, avec inlining manuel	27 s	24 Mstops

TAB. 3.2 – *Performances d'un code effectuant une décomposition LU*

L'utilisation de CADNA, même *inlinée*, est beaucoup plus pénalisante dans le premier exemple (un facteur 15 en temps de calcul) que dans le deuxième (facteur 4). Ceci s'explique par le fait que le compilateur reconnaît la multiplication matricielle et fait appel à un sous-programme interne idéalement optimisé du point de vue des caches ou de la gestion mémoire. Cette optimisation est inhibée par l'utilisation de types dérivés du fait de CADNA.

La version vectorielle de CADNA a également été testée sur le code ORCA [135] développé à l'IPSL (Institut Pierre Simon Laplace). Ce code simule l'évolution des courants océaniques pendant plusieurs milliers d'années. Il compte environ 50 000 lignes de code Fortran réparties dans 112 fichiers. Afin de pouvoir utiliser la version vectorielle de CADNA, sept procédures ont été réécrites avec la notation "tableau" de Fortran 90. Si l'on se fixe dix itérations en temps pour le test, ces procédures représentent environ 65 % du temps de calcul pour la version classique vectorielle et 57 % pour la version stochastique scalaire.

Le tableau 3.3 présente les temps d'exécution sur CRAY SV1 du code ORCA pour dix itérations en temps dans les sept procédures vectorisées.

dans les sept procédures	Temps d'exécution	Performances
code vectoriel classique	32 s	210 Mflops
CADNA, version scalaire	2 868 s	2.3 Mstops
CADNA, version vectorielle	568 s	11.8 Mstops

Tab. 3.3 – *Performances sur une partie du code ORCA*

On s'aperçoit que la performance des compilateurs est un élément essentiel pour la validation numérique sur architecture vectorielle. L'incapacité des compilateurs actuels à *inliner* entièrement les surcharges d'opérateurs vectoriels coûte un facteur 2 en temps sur CRAY SV1 (nous décrirons ci-après une situation encore plus critique).

3.2.2 Sur architecture vectorielle IEEE

Nous décrivons ici les principales étapes de l'implémentation de l'ASD sur le calculateur NEC SX5 de l'IDRIS. Les différents tests effectués depuis 1999 ont mis en évidence quelques limitations du compilateur Fortran sur ce calculateur. Certains problèmes ont été résolus par NEC, d'autres ont pu être contournés, cependant nous sommes toujours en attente de certaines corrections.

Sur NEC SX5, on dispose des quatre modes d'arrondi définis dans la norme IEEE 754. Ils se sélectionnent par l'intermédiaire des bits 34 et 35 du *Process Status Word* (PSW), un mot de 64 bits définissant le statut des processus s'exécutant sur le CPU. Comme pour les architectures séquentielles, quelques lignes écrites en assembleur permettent de manipuler ce mode d'arrondi. Plus précisément, pour pouvoir utiliser le mode d'arrondi aléatoire, trois fonctions en assembleur ont été écrites :

- *arit_plinf* qui positionne le mode d'arrondi à $+\infty$
- *arit_moinf* qui positionne le mode d'arrondi à $-\infty$
- *arit_switch* qui permet de changer le mode d'arrondi de $+\infty$ à $-\infty$ ou de $-\infty$ à $+\infty$.

En mémoire, pour accéder aux mots de huit octets consécutifs, des pas de longueur 1 doivent être effectués. Compte tenu du hardware du NEC SX5, un pas de longueur paire est très pénalisant en temps. Lorsque les tableaux stochastiques sont formés de deux tableaux en double précision, un pas de longueur 2 est nécessaire. Les performances des codes correspondants ne sont pas satisfaisantes. Lorsque les tableaux stochastiques sont formés de deux tableaux en simple précision ou de trois tableaux en double précision, le pas en mémoire est de longueur impaire. Dans la bibliothèque CADNA, les tableaux stochastiques sont définis de telle manière que la longueur du pas en mémoire soit impaire.

Lors des premiers tests effectués sur NEC SX5 en 1999, il est apparu que le compilateur Fortran ne pouvait pas vectoriser d'instruction agissant sur des variables de type dérivé. Au cours de l'année 2000, le compilateur Fortran a été modifié par NEC afin qu'il puisse vectoriser les instructions agissant sur des variables de type dérivé, à condition que la taille de ces variables soit multiple de 8 octets. Cette condition peut être respectée puisque, sur NEC SX5, la taille d'une variable stochastique est 8 octets en simple précision et 24 octets en double précision.

Une contrainte fondamentale pour cette architecture est l'impossibilité de modifier le mode d'arrondi pendant une instruction vectorielle. Afin que l'utilisation du mode d'arrondi aléatoire ne casse pas la vectorisation, plusieurs implémentations des opérateurs vectoriels ont été testées.

On considère l'opération vectorielle stochastique $C = A\Omega B$, où A, B et C sont des tableaux stochastiques de taille n. L'opérateur arithmétique ω a été surchargé afin que les éléments (flottants) des deux tableaux $C\%x$ et $C\%y$ puissent être évalués, puis arrondis aléatoirement vers $+\infty$ ou $-\infty$.

Une première implémentation consiste à calculer tous les éléments du tableau $A\%x * B\%x$ avec les deux modes d'arrondi vers $+\infty$ et $-\infty$. Les éléments de $C\%x$ sont ensuite sélectionnés grâce à la fonction *merge* du Fortran 90 et à un tableau logique L initialisé aléatoirement.

Le tableau stochastique *aux* est utilisé pour stocker les résultats obtenus avec l'un des modes d'arrondi. Avec cette implémentation, la multiplication stochastique $C = A*B$ est surchargée comme suit :

```
aux(1:n)%x=A%x*B%x
aux(1:n)%y=A%y*B%y
call arit_switch()
C%x = merge(aux%x,A%x*B%x,L(1:n))
C%y = merge(A%y*B%y,aux%y,L(1:n))
```

Comme le recommande la théorie, les éléments $C(i)\%x$ et $C(i)\%y$ $(i = 1,...,n)$ sont toujours calculés avec des modes d'arrondi opposés.

La seconde implémentation utilise la propriété suivante :
soit $u *^{+} v$ (respectivement $u *^{-} v$) le produit de deux variables flottantes u et v arrondi vers $+\infty$ (respectivement $-\infty$) :

$$u *^{+} v = -((-u) *^{-} v) \quad \text{et} \quad u *^{-} v = -((-u) *^{+} v).$$

Une multiplication dans l'un des deux modes d'arrondi peut donc être effectuée grâce à une multiplication dans l'autre mode d'arrondi. L'implémentation correspondante utilise l'instruction *where* :

```
where(L(1:n))
```

```
    C%x=A%x*B%x
    C%y=-((-A%y)*B%y)
  elsewhere
    C%y=A%y*B%y
    C%x=-((-A%x)*B%x)
  endwhere
```

Compte tenu du nombre d'opérations flottantes, avec une instruction *where* les opérations sont effectuées pour tous les éléments du tableau L. Ensuite, un masque est utilisé pour ne retenir que la moitié des résultats.

Pour 1 600 000 multiplications de tableaux stochastiques de taille 500 en simple précision, nous avons obtenu les performances suivantes.

	Temps d'exécution	Performances
Implémentation n° 1	17.9 s	45 Mstops
Implémentation n° 2	11.6 s	69 Mstops

TAB. 3.4 – *Performances pour* 1 600 000 *multiplications vectorielles en simple précision*

La seconde implémentation, pour laquelle les performances sont meilleures, a été utilisée dans la bibliothèque CADNA. Les mêmes tests ont été effectués pour l'addition stochastique vectorielle et des performances comparables ont été mesurées.

Au cours de l'an 2000, il est apparu que le compilateur ne pouvait pas *inliner* les fonctions qui renvoient un tableau. On pouvait alors remarquer un rapport voisin de 7 entre le code utilisant la bibliothèque CADNA et celui où les opérateurs stochastiques sont *inlinés* manuellement. Une correction partielle a été apportée à ce problème par NEC en 2001, afin que l'*inlining* d'une opération arithmétique entre deux tableaux de stochastiques soit possible.

Dans les fonctions de surcharge des opérateurs vectoriels, la taille des tableaux traités était déterminée automatiquement grâce à la fonction intrinsèque du Fortran "SIZE". Les performances des opérateurs stochastiques vectoriels ont à nouveau pu être améliorées grâce à l'utilisation d'une variable globale que l'on affecte à la taille des tableaux traités.

Le tableau 3.5 présente les performances obtenues lors de ces corrections successives avec un code effectuant 1 600 000 multiplications de tableaux stochastiques de taille 500 en double précision. On indique également les performances obtenues lorsque les opérateurs stochastiques sont *inlinés* manuellement.

Les performances de ce code se sont améliorées d'un facteur 6.5 depuis les tests effectués en 2000. On peut désormais constater un rapport 1.2 entre le code utilisant la bibliothèque CADNA et celui où les opérateurs stochastiques sont *inlinés* manuellement. Les mêmes tests effectués pour l'addition stochastique vectorielle ont fourni des performances similaires.

En juin 2002, il est apparu que les fonctions de surcharge d'opérateurs entre vecteurs et scalaires ne pouvaient pas être *inlinées*. Or ce type d'opération est très souvent utilisé, notamment dans les algorithmes d'algèbre linéaire. Une correction a été apportée par NEC en juin 2003.

	Temps d'exécution	Performances
Avant correction du problème d'inlining	13.0 s	62 Mstops
Après correction du problème d'inlining	3.5 s	229 Mstops
Avec variable globale (au lieu de SIZE)	2.0 s	400 Mstops
Avec variable globale et inlining manuel	1.7 s	471 Mstops

TAB. 3.5 – *Performances pour* 1 600 000 *multiplications vectorielles en double précision*

En juillet 2003, un nouveau problème d'*inlining* a été identifié : les instructions contenant plusieurs opérations entre tableaux stochastiques engendrent des performances décevantes. Par exemple, avec un code effectuant 1 600 000 fois l'addition de trois tableaux stochastiques de taille 500, on constate un rapport 9 entre les performances mesurées lorsque les opérateurs stochastiques sont surchargés et celles obtenues lorsqu'ils sont *inlinés* manuellement. Le problème ayant été signalé à NEC, nous sommes toujours en attente d'une correction.

La surcharge de tous les opérateurs stochastiques vectoriels a nécessité l'écriture de plusieurs modules. La directive de compilation IEXPAND permet d'inliner chacun de ces modules. En juillet 2003, nous avons remarqué que dans certains cas, cette directive était inefficace. En effet, avec cette directive, on constatait un rapport 400 entre les performances de certains codes et les versions où les opérateurs stochastiques étaient "inlinés" manuellement. Une correction, apportée par NEC en novembre 2003, a permis de réduire considérablement ce rapport qui vaut maintenant 1.6.

Certains codes nous posent toujours des problèmes : erreurs à la compilation ou performances décevantes. La recherche des causes précises de ces problèmes n'a pas encore abouti. Le portage de CADNA sur NEC SX5 a mis en évidence plusieurs limitations du compilateur Fortran. Parmi les différents problèmes signalés à NEC, quatre ne sont pas encore résolus.

Le tableau 3.6 présente les performances sur NEC SX5 d'un code effectuant la multiplication de deux matrices de dimension 1 000 en double précision.
Dans la version classique du code, de nouveau, la multiplication de matrices est reconnue par le compilateur. Celui-ci transforme le code afin d'effectuer cette multiplication grâce à une fonction optimisée. Avec cette fonction, la performance de crête du processeur (8 Gflops) est presque atteinte.
Comme en 3.2.1, les résultats concernant la version scalaire de CADNA ont été obtenus avec un code utilisant des boucles "DO" ; les résultats concernant la version vectorielle de CADNA ont été obtenus avec un code utilisant la notation "tableau" de Fortran 90. On remarque le facteur 358 entre les performances de la version scalaire de CADNA et celles de la version vectorielle. Le code effectuant une multiplication de matrices a aussi été écrit avec un *inlining* manuel des opérateurs stochastiques vectoriels. On remarque un facteur 2.8 entre les performances du code utilisant la version vectorielle de CADNA et celles obtenues en *inlinant* manuellement les opérateurs stochastiques vectoriels.

Le tableau 3.7 présente les performances sur NEC SX5 d'un code effectuant la décomposition LU sans pivotage d'une matrice de dimension 1 000 en double précision. On remarque le facteur 89 entre les performances de la version scalaire de CADNA et celles de la version

	Temps d'exécution	Performances
code vectoriel classique	0.26 s	7.837 Gflops
CADNA, version scalaire	3 187 s	0.64 Mstops
CADNA, version vectorielle	8.9 s	224 Mstops
CADNA, avec inlining manuel	3.2 s	631 Mstops

TAB. 3.6 – *Performances d'un code effectuant une multiplication de matrices*

vectorielle. On remarque le facteur 10 entre les performances de la version vectorielle de CADNA et celles obtenues en *inlinant* manuellement les opérateurs stochastiques vectoriels.

	Temps d'exécution	Performances
code vectoriel classique	0.55 s	1.221 Gflops
CADNA, version scalaire	1 207 s	0.56 Mstops
CADNA, version vectorielle	13.5 s	50 Mstops
CADNA, avec inlining manuel	1.3 s	501 Mstops

TAB. 3.7 – *Performances d'un code effectuant une décomposition LU*

Il est remarquable que, pour la première fois, le 1/2 Gstops a été dépassé. Nous espérons des progrès du compilateur quant à la possibilité d'*inliner* la surcharge d'opérateurs, afin que performance et validation deviennent compatibles.

Avec l'arithmétique à virgule flottante, les performances dépendent fortement du code. Par exemple, sur CRAY SV1 et NEC SX5, les performances de la multiplication de matrices, souvent effectuée grâce à une "boîte noire" interne où l'accès à la mémoire est optimisé, sont bien meilleures que celles de la décomposition LU. Le rapport peut atteindre 5 sur CRAY et 6.5 sur NEC. Avec CADNA aucune "boîte noire" interne n'est utilisée et les performances sont constantes. Ceci explique, par exemple sur NEC, un rapport 12 entre les meilleurs temps avec et sans CADNA pour la multiplication et un rapport de seulement 2.4 pour la décomposition LU.

On remarque aussi que sur les calculateurs qui respectent la norme IEEE 754, la validation numérique en ASD est moins coûteuse que sur les calculateurs non-IEEE. Par exemple, le coût en temps d'exécution de la bibliothèque CADNA pour la décomposition LU est 1.6 fois plus important sur CRAY SV1 que sur NEC SX5.

Les performances de l'ASD dépendent fortement du compilateur. Sur architecture vectorielle, seules les instructions stochastiques écrites avec la notation "tableau" de Fortran 90 sont vectorisées. Afin que les instructions qui utilisent une boucle "DO" le soient également, le développement d'un préprocesseur vectoriel qui tienne compte des tableaux stochastiques sera nécessaire.

3.3 Conclusion et perspectives

L'ASD permet, pour un coût modéré en temps d'exécution, de contrôler la qualité numérique des codes de calcul scientifique. Bien que les modifications à apporter aux codes pour son utilisation soient peu nombreuses, elles peuvent être effectuées automatiquement par un logiciel. Ces modifications ne concernent pas les méthodes numériques utilisées. En revanche, l'arithmétique d'intervalles nécessite des algorithmes spécifiques. L'ASD peut être utilisée sans étude préalable du code à contrôler. Une analyse est néanmoins nécessaire pour interpréter les instabilités numériques détectées par l'ASD.

Différentes études ont porté sur l'implémentation de l'ASD afin d'optimiser ses performances et de l'adapter à l'arithmétique à virgule flottante du processeur (qui respecte ou non la norme IEEE 754), à l'architecture (séquentielle, vectorielle ou parallèle), au compilateur (notamment à ses possibilités d'*inlining* et à sa gestion de la mémoire). Une implémentation de l'ASD a, de plus, été effectuée pour l'arithmétique à virgule fixe [43]. Outre les solutions logicielles, une architecture matérielle dédiée à l'ASD a été proposée par R. Chotin-Avot au cours de sa thèse [46].
Le coût en temps d'exécution de l'ASD est en partie dû au calcul du nombre de chiffres significatifs exacts, assuré dans le logiciel CADNA par la fonction CESTAC grâce à la formule (2.7). Une formule moins coûteuse a été utilisée par R. Chotin-Avot pour son architecture matérielle et a permis d'écrire une version plus performante de la fonction CESTAC. Une étude [16] a porté sur la validité de cette nouvelle version. En outre, a été réalisée récemment une version de CADNA qui permet, lorsque certaines variables sont plusieurs fois utilisées sans être modifiées, d'économiser une partie des appels à la fonction CESTAC [113].

De nouvelles études concernant les aspects pratiques de l'arithmétique stochastique pourront être menées. Il est en effet primordial d'adapter l'implémentation de l'ASD à l'évolution des caractéristiques matérielles et logicielles des calculateurs.
Les spécificités des processeurs cibles, qui ont été utilisées jusqu'à présent pour implanter au mieux le mode d'arrondi aléatoire, pourraient être prises en compte pour développer une version plus performante de l'ASD.
La propagation d'erreur d'arrondi sur grilles de calcul constitue un autre axe de recherche possible. L'exécution d'un code en ASD ou en arithmétique d'intervalles sur un ensemble de machines présentant différentes arithmétiques, différentes architectures et différents compilateurs est aujourd'hui un véritable défi.

Chapitre 4

Contrôle dynamique du calcul de suites

De nombreuses méthodes numériques nécessitent le calcul de suites. Citons par exemple la méthode de Newton pour l'évaluation de racines de fonctions et les méthodes de Krylov pour la résolution de systèmes linéaires. Lorsqu'une méthode numérique utilise un pas de discrétisation, ce qui est le cas des méthodes de différences finies ou d'éléments finis, une suite d'approximations peut être générée en diminuant régulièrement la valeur de ce pas. L'augmentation du nombre d'itérations a alors une conséquence bénéfique : la diminution de l'erreur de méthode; mais provoque aussi l'augmentation de l'erreur d'arrondi. Une estimation de l'erreur de méthode (aussi appelée erreur de troncature) est en général effectuée dans les bibliothèques de calcul scientifique afin de respecter un seuil fixé par l'utilisateur.

Nous proposons, dans le cadre du calcul de suites convergentes, de prendre en compte l'erreur globale, qui comprend l'erreur de méthode et l'erreur d'arrondi. Les stratégies décrites dans ce chapitre permettent de calculer dynamiquement l'itéré pour lequel l'erreur globale est minimale et de déterminer ses chiffres significatifs communs avec la limite exacte. Ces stratégies peuvent s'appliquer au calcul d'intégrales. En effet, une suite d'approximations peut être calculée en utilisant une méthode de quadrature composite, *i.e.* fondée sur des subdivisions successives du domaine d'intégration. Il est ainsi possible d'effectuer un contrôle dynamique du calcul d'intégrales par la méthode des trapèzes, de Simpson, de Romberg ou de Gauss-Legendre. En outre, nous décrivons une stratégie de calcul pour les méthodes numériques combinant différentes suites convergentes et nous montrons comment l'appliquer au contrôle dynamique du calcul d'intégrales sur un domaine infini.

4.1 Résultats théoriques sur les suites convergentes

4.1.1 Préambule

Les résultats théoriques présentés ici ont été établis pour les suites dont la vitesse de convergence est linéaire ou exponentielle. Nous rappelons donc les propriétés qui caractérisent ces deux types de vitesse de convergence.

Définition 7 *Une suite* (I_n) *converge vers* I *avec une vitesse linéaire si*

$$I_n - I = K\alpha^n + o(\alpha^n), \text{ où } K \in \mathbb{R} \text{ et } 0 < |\alpha| < 1.$$

Avec une suite à convergence linéaire, le nombre d'itérations nécessaires pour obtenir une approximation de la limite avec une décimale exacte supplémentaire est quasi-constant.

Définition 8 *Une suite* (I_n) *converge vers* I *avec une vitesse exponentielle si*

$$I_n - I = K\,\alpha^{p^n} + o(\alpha^{p^n}), \text{ où } K \in \mathbb{R},\ 0 < |\alpha| < 1 \text{ et } p > 1.$$

Avec une suite à convergence exponentielle, à chaque itération, le nombre de chiffres exacts est quasiment multiplié par p.

Les résultats théoriques présentés par la suite s'appuient sur la notion de chiffres significatifs communs à deux nombres réels. Nous rappelons donc la définition suivante.

Définition 9 *Soient* a *et* b *deux nombres réels, le nombre de chiffres significatifs communs à* a *et* b *est défini dans* $\mathbb{R}$ *par*

1. *si* $a \neq b$, $C_{a,b} = \log_{10}\left|\dfrac{a+b}{2(a-b)}\right|$,
2. $\forall a \in \mathbb{R}$, $C_{a,a} = +\infty$.

Alors $|a-b| = \left|\frac{a+b}{2}\right| 10^{-C_{a,b}}$. Par exemple, si $C_{a,b} = 3$, la différence relative entre a et b est de l'ordre de 10^{-3}, ce qui signifie que a et b ont trois chiffres significatifs en commun.

4.1.2 Sur les suites à convergence linéaire

Considérons une suite (I_n) qui converge linéairement vers I. À partir des chiffres significatifs communs à deux itérés successifs, I_n et I_{n+1}, le théorème suivant nous permet de déterminer les chiffres significatifs communs à I_n et à la limite exacte I.

Théorème 2 *Soit une suite* (I_n) *qui converge linéairement vers* I*, i.e. qui vérifie* $I_n - I = K\alpha^n + o(\alpha^n)$ *où* $K \in \mathbb{R}$ *et* $0 < |\alpha| < 1$*, alors*

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha}\right) + o\left(1\right).$$

La preuve de ce théorème, établie dans [91], peut être consultée en 8.7.

Lorsque la zone de convergence est atteinte, *i.e.* lorsque le terme $o\left(1\right)$ devient négligeable, les chiffres décimaux significatifs communs à I_n et I_{n+1} sont aussi en commun avec la limite exacte I, à $\log_{10}\left(\frac{1}{1-\alpha}\right)$ près.
Si $\alpha \in]-1{,}0[$, alors $-\log_{10} 2 < \log_{10}\left(\frac{1}{1-\alpha}\right) < 0$. Dans ce cas, si la zone de convergence est atteinte, les chiffres significatifs communs à I_n et I_{n+1} sont aussi en commun avec I.
$\forall \alpha \in]0{,}1[$, $\exists k\ 0 < \alpha \leq 1 - 10^{-k}$ et donc $0 < \log_{10}\left(\frac{1}{1-\alpha}\right) \leq k$. Si la zone de convergence est atteinte, les chiffres significatifs communs à I_n et I_{n+1} sont aussi en commun avec I, à k

près. Plus le paramètre α est faible, plus la convergence de la suite est rapide et plus k est faible.

Remarque 1 *Si $0 < \alpha \leq \frac{1}{2}$, alors $0 < \log_2\left(\frac{1}{1-\alpha}\right) \leq 1$. Dans ce cas, si la zone de convergence est atteinte, les bits significatifs communs à I_n et I_{n+1} sont aussi en commun avec I, à un près.*

4.1.3 Sur les suites à convergence exponentielle

Un résultat théorique similaire à celui du théorème 2 a été établi pour les suites à convergence exponentielle.

Théorème 3 *Soit une suite (I_n) qui converge vers I avec une vitesse exponentielle, i.e. qui vérifie $I_n - I = K\,\alpha^{p^n} + o(\alpha^{p^n})$ où $K \in \mathbb{R}$, $0 < |\alpha| < 1$ et $p > 1$, alors*

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha^{p^n(p-1)}}\right) + o\,(1)\,.$$

La preuve de ce théorème, donnée dans [91], peut être consultée en 8.7.

Si la zone de convergence est atteinte, les chiffres décimaux significatifs communs à I_n et I_{n+1} sont aussi en commun avec la limite exacte I à $\log_{10}\left(\frac{1}{1-\alpha^{p^n(p-1)}}\right)$ près.
Si $0 < |\alpha| \leq M_n$, avec $M_n = (\frac{9}{10})^{\left(\frac{1}{p^n(p-1)}\right)}$, alors $0 < \log_{10}\left(\frac{1}{1-\alpha^{p^n(p-1)}}\right) \leq 1$. Les chiffres significatifs communs à I_n et I_{n+1} sont aussi en commun avec I, à un près.
Lorsque le nombre n d'itérations augmente, M_n augmente aussi et la condition que α doit vérifier pour que $\log_{10}\left(\frac{1}{1-\alpha^{p^n(p-1)}}\right) \leq 1$ devient de moins en moins stricte. Par exemple, si la suite (I_n) a une convergence quadratique, ce qui est caractérisé par $p = 2$, alors $M_1 > 0.94$ et $M_5 > 0.99$. De même, lorsque p augmente, la suite converge plus rapidement et M_n augmente aussi.

Remarque 2 *Si la zone de convergence est atteinte, les bits significatifs communs à I_n et I_{n+1} sont aussi en commun avec la limite exacte I, à $\log_2\left(\frac{1}{1-\alpha^{p^n(p-1)}}\right)$ près. Si $0 < |\alpha| \leq 2^{\left(\frac{1}{p^n(1-p)}\right)}$, alors $0 < \log_2\left(\frac{1}{1-\alpha^{p^n(p-1)}}\right) \leq 1$. Cette condition sur α est aisément satisfaite. En effet, dans le cas d'une convergence quadratique, (i.e. pour $p = 2$) si $n = 5$, $2^{\left(\frac{1}{p^n(1-p)}\right)} > 0.97$.*

4.1.4 Sur les suites à convergence au moins linéaire

Nous considérons ici une suite (I_n) qui converge vers I au moins linéairement, *i.e.* qui vérifie $|I - I_{n+1}| \leq \alpha|I - I_n|$ avec $0 < \alpha < 1$. Nous supposons que la convergence de la suite (I_n) est monotone. Les suites à convergence alternée seront traitées en 4.1.5.

Les théorèmes 4 et 5 nous permettent, à partir de deux itérés successifs, de déterminer les premiers chiffres de la limite exacte.

Théorème 4 *Soit une suite* (I_n) *qui converge vers* I *telle que*

$$\forall n \in I\!N,\ I_n \geq 0 \ \ et \ \ 0 \leq I - I_{n+1} \leq \alpha(I - I_n)$$

ou

$$\forall n \in I\!N,\ I_n \leq 0 \ \ et \ \ 0 \leq I_{n+1} - I \leq \alpha(I_n - I),$$

avec $0 < \alpha < 1$*, alors*

$$C_{I_n,I_{n+1}} \leq C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha}\right).$$

Preuve La preuve correspondant au cas où $I_n \geq 0$ et $0 \leq I - I_{n+1} \leq \alpha(I - I_n)$ peut être consultée en 8.9 (théorème 15).

Si $0 \leq I_{n+1} - I \leq \alpha\,(I_n - I)$, alors

$$0 \leq (1-\alpha)\,(I_n - I) \leq I_n - I_{n+1}. \tag{4.1}$$

Si $\forall n \in I\!N$, $I_n \leq 0$, de la définition 9 et de l'équation (4.1), on déduit :

$$C_{I_n,I_{n+1}} = \log_{10}\left(\frac{I_n + I_{n+1}}{2(I_{n+1} - I_n)}\right) = \log_{10}\left(\frac{I_n}{I_{n+1} - I_n} + \frac{1}{2}\right). \tag{4.2}$$

De même,

$$C_{I_n,I} = \log_{10}\left(\frac{I_n}{I - I_n} + \frac{1}{2}\right). \tag{4.3}$$

De l'équation (4.1), on déduit :

$$\frac{I_n}{I_{n+1} - I_n} \leq \frac{1}{1-\alpha}\,\frac{I_n}{I - I_n}. \tag{4.4}$$

Comme $0 < \alpha < 1$,

$$\frac{I_n}{I_{n+1} - I_n} + \frac{1}{2} \leq \frac{1}{1-\alpha}\left(\frac{I_n}{I - I_n} + \frac{1}{2}\right). \tag{4.5}$$

Enfin, des équations (4.2) et (4.3), on déduit :

$$C_{I_n,I_{n+1}} \leq C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha}\right). \tag{4.6}$$

□

Les chiffres décimaux significatifs communs à I_n et I_{n+1} sont aussi en commun avec la limite exacte I, à $\log_{10}\left(\frac{1}{1-\alpha}\right)$ près. Comme dans le cas d'une convergence linéaire, si $\alpha \leq \frac{1}{2}$, alors les bits significatifs communs à I_n et I_{n+1} sont aussi en commun avec I, à un près.

Théorème 5 *Soit une suite* (I_n) *qui converge vers* I *telle que*

$$\forall n \in I\!N,\ I_n \geq 0 \ \ et \ \ 0 \leq I_{n+1} - I \leq \alpha(I_n - I)$$

ou

$$\forall n \in I\!N,\ I_n \leq 0 \quad et \quad 0 \leq I - I_{n+1} \leq \alpha(I - I_n),$$

avec $0 < \alpha < 1$, *alors*

$$C_{I_n,I_{n+1}} \leq C_{I_n,I} + \log_{10}\left(\frac{1+\alpha}{1-\alpha}\right).$$

Preuve Si $0 \leq I_{n+1} - I \leq \alpha\,(I_n - I)$, alors

$$0 \leq (1-\alpha)\,(I_n - I) \leq I_n - I_{n+1}. \tag{4.7}$$

Si $\forall n \in I\!N$, $I_n \geq 0$, de la définition 9 et de l'équation (4.7), on déduit :

$$C_{I_n,I_{n+1}} = \log_{10}\left(\frac{I_n + I_{n+1}}{2(I_n - I_{n+1})}\right). \tag{4.8}$$

De même,

$$C_{I_n,I} = \log_{10}\left(\frac{I_n + I}{2(I_n - I)}\right). \tag{4.9}$$

De l'équation (4.7), on déduit :

$$\frac{I_n}{I_n - I_{n+1}} \leq \frac{1}{1-\alpha}\ \frac{I_n}{I_n - I}. \tag{4.10}$$

et

$$\frac{I_{n+1}}{I_n - I_{n+1}} \leq \frac{\alpha I_n + (1-\alpha)I}{(1-\alpha)(I_n - I)}. \tag{4.11}$$

Des équations (4.10) et (4.11), on déduit :

$$\frac{I_n + I_{n+1}}{I_n - I_{n+1}} \leq \frac{(1+\alpha)I_n + (1-\alpha)I}{(1-\alpha)(I_n - I)}. \tag{4.12}$$

Comme $1 - \alpha < 1 + \alpha$, de l'équation (4.12), on déduit :

$$\frac{I_n + I_{n+1}}{2(I_n - I_{n+1})} \leq \frac{1+\alpha}{1-\alpha}\ \frac{I_n + I}{2(I_n - I)}. \tag{4.13}$$

Par conséquent,

$$C_{I_n,I_{n+1}} \leq C_{I_n,I} + \log_{10}\left(\frac{1+\alpha}{1-\alpha}\right). \tag{4.14}$$

Si $0 \leq I - I_{n+1} \leq \alpha(I - I_n)$, alors

$$0 \leq (1-\alpha)\,(I - I_n) \leq I_{n+1} - I_n. \tag{4.15}$$

Si $\forall n \in I\!N$, $I_n \leq 0$, de la définition 9 et de l'équation (4.15), on déduit :

$$C_{I_n,I_{n+1}} = \log_{10}\left(\frac{-(I_n + I_{n+1})}{2(I_{n+1} - I_n)}\right). \tag{4.16}$$

De même,

$$C_{I_n,I} = \log_{10}\left(\frac{-(I_n+I)}{2(I-I_n)}\right). \tag{4.17}$$

De l'équation (4.15), on déduit :

$$\frac{-I_n}{I_{n+1}-I_n} \leq \frac{1}{1-\alpha}\ \left(\frac{-I_n}{I-I_n}\right). \tag{4.18}$$

et

$$\frac{-I_{n+1}}{I_{n+1}-I_n} \leq \frac{-\left(\alpha I_n + (1-\alpha)I\right)}{(1-\alpha)(I-I_n)}. \tag{4.19}$$

Des équations (4.18) et (4.19), on déduit :

$$\frac{-(I_n+I_{n+1})}{I_{n+1}-I_n} \leq \frac{-\left((1+\alpha)I_n + (1-\alpha)I\right)}{(1-\alpha)(I-I_n)}. \tag{4.20}$$

Comme $1-\alpha < 1+\alpha$, de l'équation (4.20), on déduit :

$$\frac{-(I_n+I_{n+1})}{2(I_{n+1}-I_n)} \leq \frac{1+\alpha}{1-\alpha}\ \left(\frac{-(I_n+I)}{2(I_n-I)}\right). \tag{4.21}$$

Par conséquent,

$$C_{I_n,I_{n+1}} \leq C_{I_n,I} + \log_{10}\left(\frac{1+\alpha}{1-\alpha}\right). \tag{4.22}$$

$\square$

Les chiffres décimaux significatifs communs à I_n et I_{n+1} sont aussi en commun avec la limite exacte I, à $\log_{10}\left(\frac{1+\alpha}{1-\alpha}\right)$ près. Si $\alpha \leq \frac{1}{3}$, alors $\log_2\left(\frac{1+\alpha}{1-\alpha}\right) \leq 1$. Dans ce cas, les bits significatifs communs à I_n et I_{n+1} sont aussi en commun avec I, à un près.

4.1.5 Sur les suites à convergence alternée

Lors du calcul d'une suite à convergence alternée, le théorème 6 nous permet, à partir de deux itérés successifs, de déterminer les premiers chiffres de la limite exacte.

Théorème 6 *Soit une suite (I_n) de limite I, à convergence alternée, dont les termes sont de signe constant, alors*

$$C_{I_n,I_{n+1}} \leq C_{I_n,I}.$$

Preuve La preuve relative aux suites positives à convergence alternée peut être consultée en 8.9 (théorème 16).

Si $\forall n \in I\!N$, $I_n \leq 0$ et $0 \leq I_n - I \leq I_n - I_{n+1}$, alors

$$0 \leq \frac{-I_n}{I_n-I_{n+1}} \leq \frac{-I_n}{I_n-I_{n+1}}. \tag{4.23}$$

De la définition 9, on déduit :

$$C_{I_n,I_{n+1}} = \log_{10}\left(\frac{-(I_n+I_{n+1})}{2(I_n-I_{n+1})}\right) = \log_{10}\left(\frac{-I_n}{I_n-I_{n+1}}+\frac{1}{2}\right). \tag{4.24}$$

De même,

$$C_{I_n,I} = \log_{10}\left(\frac{-I_n}{I_n-I}+\frac{1}{2}\right). \tag{4.25}$$

Par conséquent,

$$C_{I_n,I_{n+1}} \leq C_{I_n,I}. \tag{4.26}$$

Si $\forall n \in I\!N$, $I_n \leq 0$ et $0 \leq I - I_n \leq I_{n+1} - I_n$, alors

$$0 \leq \frac{-I_n}{I_{n+1}-I_n} \leq \frac{-I_n}{I-I_n}. \tag{4.27}$$

De la définition 9, on déduit :

$$C_{I_n,I_{n+1}} = \log_{10}\left(\frac{-(I_n+I_{n+1})}{2(I_{n+1}-I_n)}\right) = \log_{10}\left(\frac{-I_n}{I_{n+1}-I_n}-\frac{1}{2}\right). \tag{4.28}$$

De même,

$$C_{I_n,I} = \log_{10}\left(\frac{-I_n}{I-I_n}-\frac{1}{2}\right). \tag{4.29}$$

Par conséquent,

$$C_{I_n,I_{n+1}} \leq C_{I_n,I}. \tag{4.30}$$

□

Les chiffres décimaux significatifs communs à I_n et I_{n+1} sont aussi en commun avec la limite exacte I.

Remarque 3 *En 4.1.4 et 4.1.5, nous avons considéré des suites dont les termes sont de signe constant. Cette hypothèse n'est en fait pas restrictive. En effet, si la limite n'est pas nulle, à partir d'un certain rang les termes sont de même signe que la limite. Si la limite est nulle et la convergence monotone, le théorème 5 peut s'appliquer : les termes sont de signe constant. Si la limite est nulle et la convergence alternée, alors la comparaison des chiffres significatifs de deux itérés successifs, I_n et I_{n+1}, présente peu d'intérêt : $C_{I_n,I_{n+1}} \leq 0$.*

4.1.6 Applications en arithmétique stochastique discrète

Les théorèmes 2 à 6 nous permettent, lorsque la zone de convergence d'une suite est atteinte, de déterminer, à partir de deux itérés successifs, les premiers chiffres significatifs de la limite exacte. Ces théorèmes ont été établis en tenant compte uniquement de l'erreur de troncature sur un itéré. Or les résultats obtenus sur ordinateur avec une arithmétique à précision finie sont entachés d'erreur d'arrondi. Pour pouvoir néanmoins appliquer ces théorèmes, il faut donc évaluer ou contrôler le bruit numérique dû à la propagation d'erreur d'arrondi, par

exemple grâce à l'arithmétique d'intervalles [6, 108] ou à l'ASD (Arithmétique Stochastique Discrète).
L'ASD nous permet d'estimer le nombre de chiffres significatifs exacts d'un résultat calculé sur ordinateur, *i.e.* ses chiffres significatifs qui ne sont pas entachés d'erreur d'arrondi. Considérons une suite (I_n) calculée en ASD et supposons que sa zone de convergence soit atteinte. Si la différence entre deux itérés successifs est non significative, *i.e.* $I_n - I_{n+1} = @.0$, alors cette différence est uniquement due aux erreurs d'arrondi. Cette condition est qualifiée de *test d'arrêt optimal* [7, 178, 179, 180, 182] : effectuer des itérations supplémentaires serait inutile. On peut donc déterminer dynamiquement, lors de l'exécution du code, l'itéré optimal, *i.e.* celui pour lequel l'erreur globale, qui comprend à la fois l'erreur de troncature et l'erreur d'arrondi, est minimale. En outre, d'après les formules établies dans les théorèmes 2 à 6, on peut déterminer quels chiffres significatifs exacts de I_{n+1} sont en commun avec I.

4.2 Contrôle dynamique des méthodes d'approximation

Certaines méthodes numériques utilisent un pas de discrétisation h et permettent alors d'approcher une valeur exacte L par une valeur $L(h)$ telle que $\lim_{h\to 0} L(h) = L$. Si le pas h décroît, l'erreur de troncature sur $L(h)$ décroît également, mais en général l'erreur d'arrondi augmente alors. Il peut donc s'avérer difficile de déterminer le pas optimal, *i.e.* celui pour lequel l'erreur globale, formée de l'erreur de troncature et de l'erreur d'arrondi, est minimale. Nous proposons une stratégie, fondée sur des divisions successives du pas par 2, qui permet de déterminer dynamiquement le pas optimal. Grâce au théorème 7, nous montrons alors que les bits significatifs exacts de l'approximation obtenue sont en commun avec le résultat exact, à un près.

Théorème 7 *Considérons une méthode numérique qui fournit une approximation $L(h)$ d'ordre p d'une valeur exacte L, i.e. $L(h) - L = Kh^p + \mathcal{O}(h^q)$ avec $1 \le p < q$, $K \in \mathbb{R}$.*
Si L_n est l'approximation calculée avec le pas $\frac{h_0}{2^n}$, alors

$$C_{L_n,L_{n+1}} = C_{L_n,L} + \log_{10}\left(\frac{2^p}{2^p-1}\right) + \mathcal{O}\left(2^{n(p-q)}\right).$$

Preuve L'erreur de troncature sur L_n est :

$$L_n - L = K\left(\frac{h_0}{2^n}\right)^p + \mathcal{O}\left(\frac{1}{2^{qn}}\right) \tag{4.31}$$

On utilisant la même formule pour L_{n+1}, on obtient :

$$L_n - L_{n+1} = K\ \left(\frac{2^p-1}{2^p}\right)\ \left(\frac{h_0}{2^n}\right)^p + \mathcal{O}\left(\frac{1}{2^{qn}}\right) \tag{4.32}$$

De l'équation (4.31), on déduit :

$$\frac{L_n}{L_n - L} = \frac{L_n}{K(\frac{h_0}{2^n})^p\,(1+\mathcal{O}(2^{n(p-q)}))} \tag{4.33}$$

$$\frac{L_n}{L_n - L} = \frac{L_n}{K(\frac{h_0}{2^n})^p}\left(1 + \mathcal{O}(2^{n(p-q)})\right) \tag{4.34}$$

Donc

$$\frac{L_n}{L_n - L} = \frac{L_n}{K(\frac{h_0}{2^n})^p} + \mathcal{O}(2^{n(2p-q)}) \tag{4.35}$$

Alors

$$\frac{L_n + L}{2(L_n - L)} = \frac{L_n}{L_n - L} - \frac{1}{2} = \frac{L_n}{K(\frac{h_0}{2^n})^p} + \mathcal{O}(2^{n(2p-q)}) \tag{4.36}$$

De même, de l'équation (4.32), on déduit :

$$\frac{L_n + L_{n+1}}{2(L_n - L_{n+1})} = \frac{L_n}{L_n - L_{n+1}} - \frac{1}{2} = \left(\frac{L_n}{K(\frac{h_0}{2^n})^p}\right)\left(\frac{2^p}{2^p - 1}\right) + \mathcal{O}(2^{n(2p-q)}) \tag{4.37}$$

De la définition 9 et de l'équation (4.36) on déduit :

$$C_{L_n,L} = \log_{10}\left|\frac{L_n}{K(\frac{h_0}{2^n})^p}\left(1 + \mathcal{O}(2^{n(p-q)})\right)\right| \tag{4.38}$$

$$C_{L_n,L} = \log_{10}\left|\frac{L_n}{K(\frac{h_0}{2^n})^p}\right| + \log_{10}\left|1 + \mathcal{O}(2^{n(p-q)})\right| \tag{4.39}$$

Donc

$$C_{L_n,L} = \log_{10}\left|\frac{L_n}{K(\frac{h_0}{2^n})^p}\right| + \mathcal{O}\left(2^{n(p-q)}\right) \tag{4.40}$$

De même, de la définition 9 et de l'équation (4.37) on déduit :

$$C_{L_n,L_{n+1}} = \log_{10}\left|\left(\frac{L_n}{K(\frac{h_0}{2^n})^p}\right)\left(\frac{2^p}{2^p - 1}\right)\right| + \mathcal{O}\left(2^{n(p-q)}\right) \tag{4.41}$$

Enfin

$$C_{L_n,L_{n+1}} = C_{L_n,L} + \log_{10}\left(\frac{2^p}{2^p - 1}\right) + \mathcal{O}\left(2^{n(p-q)}\right). \tag{4.42}$$

□

Lorsque la zone de convergence est atteinte, *i.e.* lorsque le terme $\mathcal{O}\left(2^{n(p-q)}\right)$ devient négligeable, les chiffres significatifs communs à deux approximations successives L_n et L_{n+1} sont aussi en commun avec le résultat exact L, à un bit près. En effet, le terme $\log_{10}\left(\frac{2^p}{2^p-1}\right)$ décroît lorsque p augmente et il correspond à un bit pour les méthodes d'ordre 1.
On peut déterminer dynamiquement l'approximation optimale en effectuant les calculs jusqu'à ce que la différence $L_n - L_{n+1}$ soit non significative. Si la zone de convergence a bien été atteinte, les bits significatifs exacts de la dernière approximation sont alors communs avec L, à un près.

4.3 Contrôle dynamique des méthodes de quadrature

Les bibliothèques de calcul d'intégrales, telle QUADPACK [155], prennent généralement en compte l'erreur de troncature générée. Les méthodes de quadrature développées en arithmétique d'intervalles, en tenant compte à la fois de l'erreur de troncature et de l'erreur d'arrondi commises, fournissent des bornes garanties du résultat exact. Bien que les premières études sur le calcul d'intégrales en arithmétique d'intervalles [56, 58] portaient principalement sur les formules de Newton-Côtes, les procédures actuelles utilisent la méthode de Romberg ou de Gauss. En arithmétique d'intervalles, ces deux méthodes de quadrature sont fondées sur une approximation de l'intégrale par un produit scalaire et un calcul du terme d'erreur grâce à des algorithmes de différentiation automatique [78, 106, 159]. La méthode de Romberg en arithmétique d'intervalles consiste alors, non pas à calculer un tableau triangulaire de valeurs, mais à effectuer directement un produit scalaire [97]. La méthode de Gauss en arithmétique d'intervalles nécessite un encadrement des poids et des points utilisés [170]. Une stratégie adaptative contrôle la subdivision de l'intervalle d'intégration en sous-intervalles sur lesquels la procédure d'intégration est appliquée [97, 107, 171]. Cette stratégie mène généralement à un encadrement de l'intégrale dont l'erreur relative est inférieure à une borne spécifiée par l'utilisateur.

Les stratégies décrites dans cette partie permettent d'effectuer un contrôle dynamique de différentes méthodes de quadrature. Elles sont toutes fondées sur le calcul d'une suite d'approximations qui converge vers l'intégrale. En tenant compte de l'erreur de troncature commise, il est alors possible de déterminer, à partir de deux approximations successives, les premiers chiffres exacts de l'intégrale. En estimant l'erreur d'arrondi générée, on peut en outre effectuer un nombre optimal d'itérations et déterminer quels chiffres significatifs du résultat sont en commun avec la valeur exacte de l'intégrale.

4.3.1 Méthodes des trapèzes et de Simpson

Comme pour les autres méthodes d'approximation, le contrôle dynamique de la méthode des trapèzes ou de Simpson est fondé sur des divisions successives du pas d'intégration par 2. On obtient alors une suite d'approximations qui converge linéairement vers l'intégrale. Les théorèmes 8 et 9 permettent de déterminer, à partir de deux approximations successives, les premiers chiffres de la valeur exacte de l'intégrale.

Théorème 8 *Supposons que la fonction f soit C^k sur l'intervalle $[a,b]$ avec $k \geq 3$ et que $f'(b) \neq f'(a)$. Soit I_n l'approximation de $I = \int_a^b f(x)dx$ par la méthode des trapèzes avec le pas $h = \frac{b-a}{2^n}$. Alors*

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{4}{3}\right) + \mathcal{O}\left(\frac{1}{4^n}\right). \tag{4.43}$$

Théorème 9 *Supposons que la fonction f soit C^k sur l'intervalle $[a,b]$ avec $k \geq 5$ et que $f^{(3)}(b) \neq f^{(3)}(a)$. Soit I_n l'approximation de $I = \int_a^b f(x)dx$ par la méthode de Simpson avec le pas $h = \frac{b-a}{2^n}$. Alors*

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{16}{15}\right) + \mathcal{O}\left(\frac{1}{4^n}\right). \tag{4.44}$$

Les preuves de ces deux théorèmes, établies en [44], peuvent être consultées en 8.6.

Lorsque la zone de convergence est atteinte, *i.e.* lorsque le terme $\mathcal{O}\left(\frac{1}{4^n}\right)$ devient négligeable, les chiffres significatifs communs à deux approximations successives I_n et I_{n+1} sont aussi en commun avec I, la valeur exacte de l'intégrale, à un bit près. En effet, les termes $\log_{10}\left(\frac{4}{3}\right)$ et $\log_{10}\left(\frac{16}{15}\right)$ correspondent au plus à un bit.
Si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux approximations successives soit non significative, alors les bits non entachés d'erreur d'arrondi de la dernière approximation sont communs avec la valeur exacte de l'intégrale, à un près.

Les théorèmes 8 et 9 ont été établis à partir du développement de l'erreur de troncature sur I_n [168],

- pour la méthode des trapèzes :

$$I_n - I = \frac{h^2}{12}\,[f'(b) - f'(a)] + \mathcal{O}(h^4) \tag{4.45}$$

- et pour la méthode de Simpson :

$$I_n - I = \frac{h^4}{180}\,[f^{(3)}(b) - f^{(3)}(a)] + \mathcal{O}(h^6) \tag{4.46}$$

avec $h = \dfrac{b-a}{2^n}$.

On retrouve alors l'équation (4.31) qui caractérise les méthodes d'approximation, avec $p = 2$ et $q = 4$ pour la méthode des trapèzes; $p = 4$ et $q = 6$ pour la méthode de Simpson. Ceci explique la cohérence des théorèmes 8 et 9 avec le théorème 7.

4.3.2 Méthode de Gauss-Legendre

Une stratégie de calcul

L'approximation de $\int_{-1}^{1} f(x)dx$ par la méthode de Gauss-Legendre à ν points [48, 65] est $\sum_{i=1}^{\nu} C_i f(x_i)$, où pour $i = 1,...,\nu$, $\{x_i\}$ sont les racines du polynôme de Legendre de degré ν, noté P_ν, et

$$C_i = \frac{2}{(1 - x_i^2)(P'_\nu(x_i))^2}. \tag{4.47}$$

Pour le calcul d'une intégrale sur un autre intervalle telle $I = \int_a^b g(t)dt$, le changement de variable ci-après est nécessaire.

$$I = \int_a^b g(t)dt = \frac{b-a}{2}\int_{-1}^{1} f(x)dx, \tag{4.48}$$

où

$$\forall x \in [-1,1],\, f(x) = g\left(\frac{(b-a)x + b + a}{2}\right). \tag{4.49}$$

La méthode de Gauss-Legendre à ν points est d'ordre 2ν : elle fournit la valeur exacte de l'intégrale d'un polynôme de degré r avec $r \leq 2\nu - 1$.

Nous présentons une stratégie de calcul qui permet d'effectuer un contrôle dynamique de la méthode de Gauss-Legendre, bien que celle-ci n'utilise pas de pas d'intégration. Cette stratégie consiste à découper l'intervalle d'intégration en sous-intervalles de même longueur sur lesquels on applique la méthode classique de Gauss-Legendre. On construit donc une méthode d'approximation fondée sur la méthode de Gauss-Legendre.
Supposons que l'intervalle d'intégration soit partitionné en 2^n sous-intervalles et que l'intégrale sur chaque sous-intervalle soit évaluée par la méthode de Gauss-Legendre à ν points. Le théorème 10 présente l'erreur de troncature sur I_n, la somme des 2^n approximations obtenues.

Théorème 10 *Supposons que la fonction g soit C^p sur l'intervalle $[a,b]$ avec $p \geq 2\nu$.
Soit $I = \int_a^b g(t)dt$ et pour $i = 1,...,\nu$ soient $\{x_i\}$ les racines du polynôme de Legendre de degré ν et $\{C_i\}$ les poids associés.
Supposons que l'intégrale sur chaque sous-intervalle $[a + (k-1)\frac{b-a}{2^n}, a + k\frac{b-a}{2^n}]$ $(k = 1,...,2^n)$ soit calculée par la méthode de Gauss-Legendre à ν points.
Soit I_n la somme des 2^n approximations obtenues.
Alors*

$$I_n - I = \frac{K_\nu}{4^{n\nu}} + \mathcal{O}\left(\frac{1}{2^{n(2\nu+1)}}\right)$$

où $K_\nu = \frac{(b-a)^{2\nu}}{2^{2\nu+1}(2\nu)!}\left(\sum_{i=1}^{\nu} C_i x_i^{2\nu} - \frac{2}{2\nu+1}\right)\left[g^{(2\nu-1)}(b) - g^{(2\nu-1)}(a)\right]$.

La preuve du théorème 10 figure en 8.9 sous une forme plus générale où l'on effectue q partitions de l'intervalle d'intégration. Le choix d'effectuer un nombre de partitions qui ne soit pas une puissance de 2 sera discuté en 4.5.1.

Le théorème 11 permet à partir des chiffres significatifs communs à deux approximations successives, I_n et I_{n+1}, de déterminer les chiffres significatifs communs à I_n et I, la valeur exacte de l'intégrale. Nous utilisons les mêmes notations et hypothèses que pour le théorème 10.

Théorème 11

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{4^\nu}{4^\nu - 1}\right) + \mathcal{O}\left(\frac{1}{2^n}\right).$$

La preuve du théorème 11 peut être consultée en 8.9, sous une forme plus générale où un itéré est obtenu par q partitions de l'intervalle d'intégration et l'itéré suivant par $q + s$ partitions. Le théorème 11 correspond alors à $q = s = 2^n$.

Lorsque la zone de convergence est atteinte, *i.e.* lorsque le terme $\mathcal{O}\left(\frac{1}{2^n}\right)$ est négligeable, les chiffres significatifs communs à deux approximations successives sont aussi en commun avec la valeur exacte de l'intégrale, à un bit près. En effet, quel que soit ν, le terme $\log_{10}\left(\frac{4^\nu}{4^\nu-1}\right)$ correspond au plus à un bit.
Si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux approximations successives soit non significative, alors les bits non entachés d'erreur d'arrondi de la dernière approximation sont communs avec la valeur exacte de l'intégrale, à un près.

Cohérence avec les résultats précédents

L'erreur de troncature établie dans le théorème 10 est cohérente avec l'équation (4.31) qui caractérise les méthodes d'approximation, avec $p = 2\nu$, $q = 2\nu + 1$ et $h_0 = b - a$. Le théorème 11 aurait donc pu être établi à partir du théorème 7, en posant $p = 2\nu$ et $q = 2\nu+1$.

Les stratégies proposées pour le contrôle dynamique des méthodes des trapèzes, de Simpson et de Gauss-Legendre sont similaires. Pour l'évaluation de l'intégrale $I = \int_a^b g(t)dt$, on calcule une suite d'approximations (I_n) qui converge linéairement vers I.
Le contrôle dynamique de la méthode de Gauss-Legendre est fondé sur un découpage de l'intervalle $[a,b]$ en 2^n sous-intervalles sur lesquels on applique la méthode classique de Gauss-Legendre.
Avec les versions dites "composites" ou "étendues" [140] des méthodes des trapèzes ou de Simpson, I_n est l'approximation obtenue avec le pas $\frac{b-a}{2^n}$.
On peut considérer que le contrôle dynamique de la méthode des trapèzes est fondé sur un découpage de l'intervalle $[a,b]$ en 2^n sous-intervalles sur lesquels on applique la méthode des trapèzes à deux points.
Concernant la méthode de Simpson, on peut considérer que l'approximation I_n est obtenue en découpant l'intervalle $[a,b]$ en 2^{n-1} sous-intervalles sur lesquels on applique la méthode de Simpson à trois points.

4.3.3 Méthode de Romberg

La méthode de Romberg [19, 26, 168] utilise une extrapolation de Richardson à partir de résultats obtenus par la méthode des trapèzes. Plus précisément, pour l'évaluation de $I = \int_a^b f(x)dx$, la méthode de Romberg consiste à calculer les valeurs suivantes, avec $h = \frac{b-a}{M}$ $(M \geq 1)$.

$$\begin{array}{ccccc}
T_1(h) & T_1(\frac{h}{2}) & .... & T_1(\frac{h}{2^{n-3}}) & T_1(\frac{h}{2^{n-2}}) & T_1(\frac{h}{2^{n-1}}) \\
T_2(h) & T_2(\frac{h}{2}) & .... & T_2(\frac{h}{2^{n-3}}) & T_2(\frac{h}{2^{n-2}}) & \\
T_3(h) & T_3(\frac{h}{2}) & .... & T_3(\frac{h}{2^{n-3}}) & & \\
\vdots & \vdots & & & & \\
T_{n-1}(h) & T_{n-1}(\frac{h}{2}) & & & & \\
T_n(h) & & & & &
\end{array}$$

La première ligne du tableau est formée des approximations successives de I par la méthode des trapèzes avec les pas $\frac{h}{2^j}(j = 0,...,n-1)$.
Les autres lignes du tableau sont obtenues grâce à la formule suivante :
pour $p = 2,...,n$ et $j = 0,...,n-p$,

$$T_p\left(\frac{h}{2^j}\right) = \frac{1}{4^{p-1}-1}\left(4^{p-1}T_{p-1}\left(\frac{h}{2^j+1}\right) - T_{p-1}\left(\frac{h}{2^j}\right)\right) \tag{4.50}$$

La suite des valeurs trapézoïdales $T_1(h)$, $T_1(\frac{h}{2})$, ...,$T_1(\frac{h}{2^{n-1}})$ converge linéairement vers I. La suite des valeurs extrapolées $T_1(h)$, $T_2(h)$,..., $T_n(h)$ converge encore bien plus rapidement vers I.

Le théorème suivant permet de déterminer à partir de deux approximations successives, $T_n(h)$ et $T_{n+1}(h)$, les premiers chiffres de la valeur exacte de l'intégrale.

Théorème 12 *Supposons que la fonction f soit C^k sur $[a,b]$ avec $k \geq 2n+1$ et que $f^{(2n-1)}(b) \neq f^{(2n-1)}(a)$. Soit $T_n(h)$ l'approximation de $I = \int_a^b f(x)dx$ obtenue par n itérations de la méthode de Romberg avec le pas initial $h = \frac{b-a}{M}$ $(M \geq 1)$. Alors*

$$C_{T_n(h),T_{n+1}(h)} = C_{T_n(h),I} + O\left(\frac{1}{n^2}\right).$$

La preuve de ce théorème, établie en [95], peut être consultée en 8.8.

Lorsque la zone de convergence est atteinte, *i.e.* lorsque $O\left(1/n^2\right) \ll 1$, les chiffres significatifs communs à deux approximations successives, $T_n(h)$ et $T_{n+1}(h)$, sont aussi communs à la valeur exacte de l'intégrale.
Si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux approximations successives soit non significative, alors les chiffres non entachés d'erreur d'arrondi de la dernière approximation sont ceux de la valeur exacte de l'intégrale.

L'équation établie dans le théorème 12 est différente de celle du théorème 7, relative aux méthodes d'approximation. En effet, la méthode de Romberg n'est pas une méthode d'approximation dont l'ordre est connu a priori. L'approximation $T_n(h)$ obtenue par n itérations de la méthode de Romberg est la valeur exacte de l'intégrale d'un polynôme de degré r avec $r \leq 2n-1$. De plus, le contrôle dynamique de la méthode de Romberg ne consiste pas en des divisions successives d'un pas d'intégration. Ce contrôle dynamique est fondé sur la comparaison de deux approximations successives, $T_n(h)$ et $T_{n+1}(h)$, et non sur la comparaison de $T_n(h)$ et $T_n\left(\frac{h}{2}\right)$.

4.3.4 Expérimentation numérique : calcul d'une intégrale définie

Considérons l'intégrale

$$I = \int_0^1 \frac{\arctan(\sqrt{2+t^2})}{(1+t^2)\sqrt{2+t^2}}dt.$$

L'évaluation de cette intégrale est un problème qui a été posé dans [5]. D.H. Bailey et X.S. Li ont indiqué dans [17] sa valeur exacte : $I = \frac{5\pi^2}{96}$. Ses seize premiers chiffres exacts sont donc : $I \approx 0.5140418958900708$.

Cette intégrale a été évaluée en utilisant les différentes stratégies de calcul décrites en 4.3. On note I_n l'approximation de I calculée :

- en utilisant la méthode des trapèzes ou de Simpson avec le pas $\frac{1}{2^n}$,
- en partitionnant l'intervalle $[0,1]$ en 2^n sous-intervalles sur lesquels on applique la méthode de Gauss-Legendre à 12 points
- ou en effectuant n itérations de la méthode de Romberg avec le pas initial $h = 1$.

Les approximations I_n ont été calculées jusqu'à ce que la différence $I_n - I_{n+1}$ soit non significative. D'après les résultats théoriques décrits en 4.3, quelle que soit la méthode de quadrature choisie, les bits significatifs exacts (*i.e.* non entachés d'erreur d'arrondi) de la dernière approximation I_N sont communs avec I, à un près.

Le tableau 4.1 présente les approximations de I obtenues en simple et en double précision. Pour chaque suite, seuls sont indiqués les chiffres significatifs exacts du dernier itéré, estimés par l'ASD.

méthode	en simple précision	en double précision
trapèzes	$I_8 = 0.51404E+00$	$I_{19} = 0.5140418958899E+000$
Simpson	$I_8 = 0.514041E+00$	$I_{10} = 0.51404189589007E+000$
Romberg	$I_8 = 0.514041E+00$	$I_8 = 0.514041895890070E+000$
Gauss-Legendre	$I_2 = 0.5140419E+00$	$I_2 = 0.514041895890065E+000$

TAB. 4.1 – *Approximations de* $I \approx 0.5140418958900708$

On remarque que les chiffres significatifs exacts de chaque approximation I_N obtenue sont en commun avec I, à un près. L'erreur $I_N - I$ est toujours non significative : du fait des erreurs d'arrondi, l'ordinateur ne peut faire la distinction entre l'approximation obtenue et la valeur exacte de l'intégrale.
Le nombre d'itérations nécessaires pour que le test d'arrêt soit satisfait peut dépendre de la précision choisie, mais aussi de la méthode de quadrature utilisée. En effet, la vitesse de convergence de la suite mise en œuvre et la qualité numérique du résultat obtenu varient selon la méthode de quadrature. La suite générée par la méthode de Gauss-Legendre à 12 points converge particulièrement vite : en deux itérations, on obtient un résultat d'excellente qualité numérique quelle que soit la précision choisie.

4.4 Contrôle dynamique de combinaisons de suites

Les stratégies précédemment décrites peuvent être étendues à l'approximation de la limite d'une suite où chaque itéré est lui-même la limite d'une autre suite. Grâce à l'ASD, nous pouvons obtenir l'itéré optimal pour chaque suite calculée. Les résultats théoriques présentés précédemment peuvent alors être combinés pour déterminer les chiffres significatifs de l'approximation obtenue qui sont en commun avec la limite exacte. Dans ce qui suit, on désigne par des lettres majuscules les valeurs calculées en ASD et par des lettres minuscules les valeurs exactes correspondantes.

4.4.1 Une stratégie de calcul

Considérons une suite pour laquelle chaque terme u_m est la limite d'une autre suite. Plus précisément, soit (u_m) une suite qui converge au moins linéairement vers u et, pour tout m, soit $(u_{m,n})$ une suite qui converge au moins linéairement vers u_m.
Pour tout m, soit U_m l'approximation de u_m calculée en ASD. Cette approximation est obtenue en calculant la suite $(u_{m,n})$ jusqu'à ce que, dans la zone de convergence, la différence entre deux itérés successifs soit non significative.

Comme pour tout m, la suite $(u_{m,n})$ converge au moins linéairement vers u_m, d'après le théorème 2, on peut déterminer la valeur q telle que les bits significatifs exacts de U_m soient en commun avec u_m, à q près.
La figure 4.1 représente les bits significatifs de U_m et U_{m+1} lorsque la différence $U_m - U_{m+1}$ est non significative. Dans ce cas, les bits significatifs exacts de U_{m+1} sont en commun avec U_m et sont aussi en commun avec u_m et u_{m+1}, à q près.

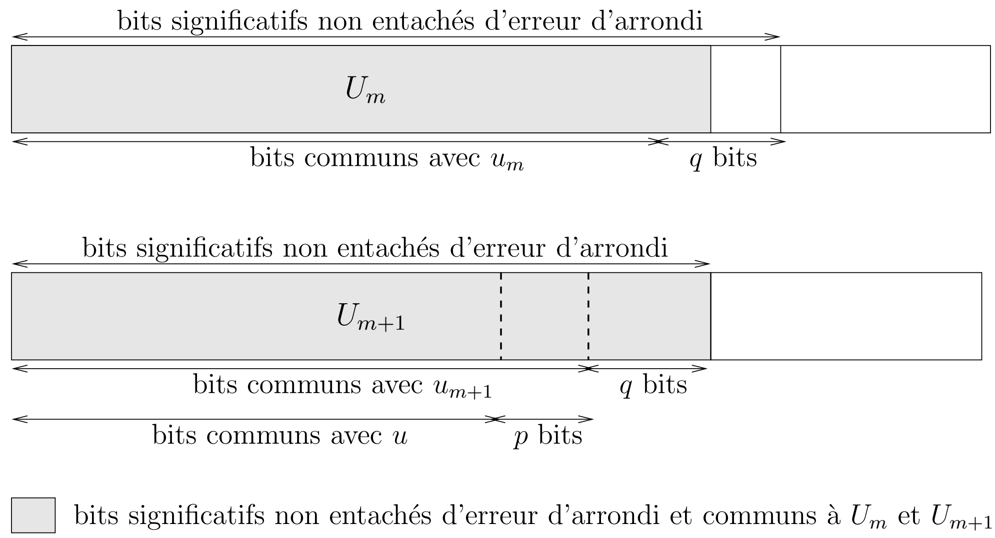

FIG. 4.1 – *Bits significatifs de U_m et U_{m+1}*

Comme la suite (u_m) converge au moins linéairement vers u, on peut déterminer la valeur p telle que les bits significatifs exacts de u_m et u_{m+1} soient en commun avec u, à p près.
Par conséquent, si la différence $U_m - U_{m+1}$ est non significative, alors les bits significatifs exacts de U_{m+1} sont en commun avec u, à $p + q$ près.

4.4.2 Contrôle dynamique du calcul d'intégrales sur un domaine infini

Une méthode pour calculer une intégrale sur un domaine infini consiste à l'évaluer sur un domaine fini suffisamment large. La stratégie de calcul décrite ci-après permet de déterminer dynamiquement la taille optimale de ce domaine fini et d'estimer le nombre de chiffres significatifs du résultat qui sont en commun avec la valeur exacte de l'intégrale.

Considérons le calcul d'une intégrale impropre $g = \int_0^\infty \phi(x)dx$. Le domaine d'intégration infini est partitionné en intervalles finis de longueur L. Soit $f_j = \int_{jL}^{(j+1)L} \phi(x)dx$ et $g_m = \sum_{j=0}^{m} f_j$, $\lim_{m\to\infty} g_m = g$.

La valeur exacte g peut être approchée numériquement par un itéré g_m, l'indice m étant suffisamment élevé. Le nombre optimal d'itérés peut être déterminé dynamiquement en effectuant les calculs en ASD.

Chaque intégrale f_j peut être évaluée en utilisant l'une des stratégies de calcul décrites en 4.3. Soit $F_{j,n}$ l'approximation de f_j calculée :

- en utilisant la méthode des trapèzes ou de Simpson avec le pas $\frac{L}{2^n}$,
- en partitionnant l'intervalle d'intégration en 2^n sous-intervalles sur lesquels on applique la méthode de Gauss-Legendre
- ou en effectuant n itérations de la méthode de Romberg.

Quel que soit j, la suite $(F_{j,n})$ est calculée jusqu'à ce que la différence entre deux itérés successifs soit non significative. Cette condition n'est pas satisfaite à la même itération pour toutes les valeurs de j. Soit n_j l'itération pour laquelle la différence $F_{j,n_j-1} - F_{j,n_j}$ est non significative. D'après les résultats théoriques décrits en 4.3, pour tout j, les bits significatifs exacts de F_{j,n_j} sont ceux de f_j, à un près.

Soit $G_m = \sum_{j=0}^{m} F_{j,n_j}$. D'après le théorème 1, les bits significatifs exacts de G_m sont ceux de g_m, à un près.

La figure 4.2 représente les bits significatifs de G_m et G_{m+1} lorsque la différence $G_m - G_{m+1}$ est non significative. Dans ce cas, les bits significatifs de G_{m+1} sont communs avec G_m et sont aussi communs avec g_m et g_{m+1}, à un près.

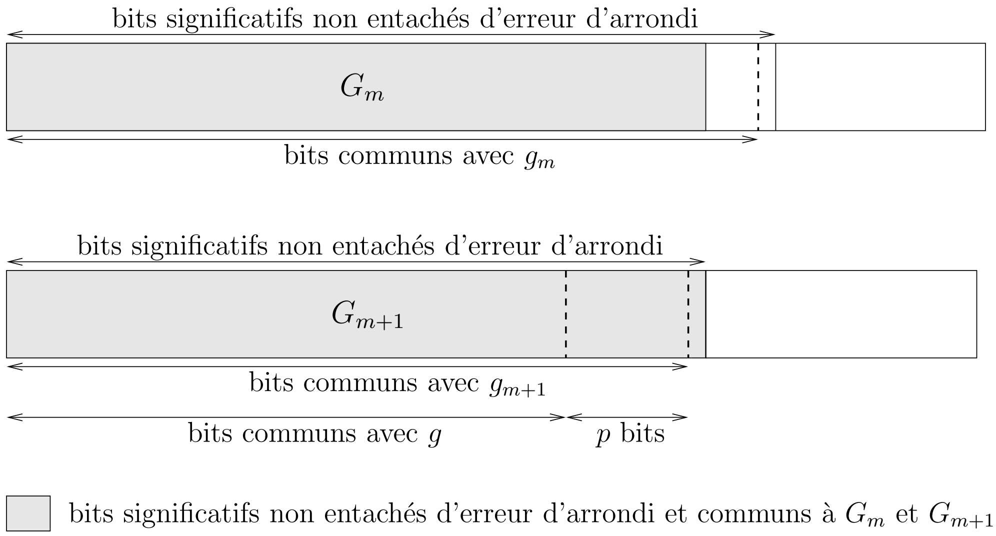

FIG. 4.2 – *Bits significatifs de G_m et G_{m+1}*

Supposons que la suite (g_m) converge au moins linéairement vers g. D'après le théorème 2, si la phase de convergence est atteinte, $C_{g_m,g_{m+1}} = C_{g_m,g} + \delta$ où δ représente p bits. Donc les bits communs à g_m et g_{m+1} sont communs avec g, à p près.
Par conséquent, si la différence $G_m - G_{m+1}$ est non significative, les bits significatifs exacts de G_{m+1} sont en commun avec g, à $p + 1$ près.

Remarque 4 *Si l'on choisit la méthode de Romberg, d'après le théorème 12, la restriction "à un bit près" s'élimine. Pour chaque intégrale calculée, les bits significatifs exacts (i.e. non entachés d'erreur d'arrondi) de l'approximation obtenue sont en commun avec la valeur mathématique correspondante. Dans ce cas, les bits significatifs exacts de l'approximation*

G_{m+1} telle que la différence $G_m - G_{m+1}$ soit non significative sont en commun avec g, à p près.

4.4.3 Expérimentation numérique : calcul d'une intégrale impropre

Considérons l'intégrale impropre

$$g = \int_0^\infty e^{-ax}\, dx = \frac{1}{a}, \text{ avec } a > 0.$$

L'intégrale g a été évaluée en utilisant la stratégie décrite en 4.4.2. En adoptant les mêmes notations qu'en 4.4.2, soit $g_m = \sum_{j=0}^{m} f_j$, où $f_j = \int_{jL}^{(j+1)L} e^{-ax}\, dx$. Les approximations des intégrales f_j ont été calculées avec la méthode de Simpson en ASD. Pour chaque valeur de j, une suite a été calculée jusqu'à ce que la différence entre deux itérés successifs soit non significative.
Comme $g_m - g = \int_{(m+1)L}^{\infty} e^{-ax}\, dx = \frac{\alpha^{m+1}}{a}$, où $\alpha = e^{-aL}$, la suite (g_m) converge linéairement vers g. On peut donc appliquer le théorème 2 : si la zone de convergence est atteinte, les bits significatifs communs à deux itérés successifs sont aussi en commun avec g, à $\log_2(\frac{1}{1-\alpha})$ près.
Soit G_m l'approximation de g_m obtenue en ASD. La suite (G_m) a été calculée jusqu'à ce que la différence entre deux itérés successifs soit non significative. Notons M le numéro de l'itération à laquelle ce test d'arrêt est satisfait : $G_{M-1} - G_M = @.0$.
D'après les résultats décrits en 4.4.2, les bits significatifs exacts de G_M sont en commun avec g, à $\log_2(\frac{1}{1-\alpha}) + 1$ près. Les chiffres décimaux significatifs exacts de G_M sont donc en commun avec g, à δ près, avec $\delta = \log_{10}(\frac{2}{1-\alpha})$.

Le tableau 4.2 présente pour $a = 1$ et différentes valeurs de L les approximations G_M obtenues en double précision. δ est une approximation du nombre de chiffres significatifs exacts de G_M qui ne sont pas en commun avec g. Lorsque la longueur L augmente, le nombre M d'intégrales f_j à calculer décroît.
Seuls sont indiqués les chiffres significatifs exacts de G_M : les autres chiffres significatifs sont entachés d'erreur d'arrondi. On remarque que le nombre de chiffres significatifs exacts obtenus (de treize à quinze) est satisfaisant pour des calculs effectués en double précision. Les chiffres significatifs exacts qui ne sont pas en commun avec la valeur exacte $g = 1$ peuvent être facilement identifiés. Par exemple, si $L = 10^{-1}$, sur les quatorze chiffres significatifs exacts de G_M, les deux derniers ne sont pas en commun avec g.
On remarque que, pour chaque approximation G_M reportée dans le tableau 4.2, ses chiffres significatifs exacts sont en commun avec g, à $\lceil\delta\rceil$ près.

Le tableau 4.3 présente pour $a = 10^{-5}$ et différentes valeurs de L les chiffres significatifs exacts des approximations G_M obtenues en double précision. Comme dans le tableau 4.2, on remarque que si la longueur L augmente, le nombre M d'intégrales f_j à calculer décroît. Pour chaque approximation G_M obtenue, on peut facilement identifier ses chiffres significatifs exacts qui sont en commun avec la valeur exacte $g = 10^5$. Comme dans le tableau 4.2, on remarque que les chiffres significatifs exacts de G_M sont communs avec g, à $\lceil\delta\rceil$ près.

L	$\delta \approx$	M	G_M
10^{-2}	2.3	2335	0.9999999999276E+000
10^{-1}	1.3	284	0.99999999999953E+000
1	0.5	33	0.999999999999996E+000
10	0.3	4	0.99999999999999E+000
50	0.3	2	0.10000000000004E+001

TAB. 4.2 – *Résultats obtenus avec la méthode de Simpson pour $a = 1$*

L	$\delta \approx$	M	G_M
10^2	3.3	19136	0.999999995109E+005
10^3	2.3	2346	0.9999999999352E+005
10^4	1.3	279	0.99999999999923E+005
10^5	0.5	33	0.999999999999995E+005
10^6	0.3	5	0.99999999999999E+005

TAB. 4.3 – *Résultats obtenus avec la méthode de Simpson pour $a = 10^{-5}$*

4.5 Stratégies alternatives

Quels résultats peut-on obtenir si l'on ne respecte pas strictement les stratégies établies précédemment ? La plupart d'entre elles sont fondées sur la progression régulière d'un pas, divisé par 2 pour chaque nouvelle approximation. Dans cette partie, nous proposons d'autres réductions du pas, qui ne sont pas nécessairement régulières.

4.5.1 Sur le choix du pas

Un contrôle dynamique basé sur des divisions du pas par 2 peut s'avérer parfois coûteux. Quelles conséquences peut avoir l'utilisation d'un pas qui ne soit pas de la forme $\frac{h_0}{2^n}$? Le théorème 13 s'applique lorsqu'une méthode d'approximation d'ordre p est utilisée avec le pas $\frac{h_0}{n}$.

Théorème 13 *Considérons une méthode numérique qui fournit une approximation $L(h)$ d'ordre p d'une valeur exacte L, i.e. $L(h) - L = Kh^p + \mathcal{O}(h^q)$ avec $1 \leq p < q$, $K \in \mathbb{R}$. Si L_n est l'approximation calculée avec le pas $\frac{h_0}{n}$, alors*

$$C_{L_n,L_{n+1}} = C_{L_n,L} + \log_{10}\left(\frac{1}{1 - (\frac{n}{n+1})^p}\right) + \mathcal{O}\left(n^{p-q}\right).$$

Preuve L'erreur de troncature sur L_n est :

$$L_n - L = K\left(\frac{h_0}{n}\right)^p + \mathcal{O}\left(\frac{1}{n^q}\right) \tag{4.51}$$

On utilisant la même formule pour L_{n+1}, on obtient :

$$L_n - L_{n+1} = K\left(1 - \left(\frac{n}{n+1}\right)^p\right)\left(\frac{h_0}{n}\right)^p + \mathcal{O}\left(\frac{1}{n^q}\right) \tag{4.52}$$

De l'équation (4.51), on déduit :

$$\frac{L_n}{L_n - L} = \frac{L_n}{K(\frac{h_0}{n})^p} + \mathcal{O}(n^{2p-q}) \tag{4.53}$$

Alors

$$\frac{L_n + L}{2(L_n - L)} = \frac{L_n}{L_n - L} - \frac{1}{2} = \frac{L_n}{K(\frac{h_0}{n})^p} + \mathcal{O}(n^{2p-q}) \tag{4.54}$$

De même, de l'équation (4.52), on déduit :

$$\frac{L_n + L_{n+1}}{2(L_n - L_{n+1})} = \frac{L_n}{L_n - L_{n+1}} - \frac{1}{2} = \left(\frac{L_n}{K(\frac{h_0}{n})^p}\right) \left(\frac{1}{1 - (\frac{n}{n+1})^p}\right) + \mathcal{O}(n^{2p-q}) \tag{4.55}$$

De la définition 9 et de l'équation (4.54) on déduit :

$$C_{L_n,L} = \log_{10} \left| \frac{L_n}{K(\frac{h_0}{n})^p} \right| + \mathcal{O}\left(n^{p-q}\right) \tag{4.56}$$

De même, de la définition 9 et de l'équation (4.55) on déduit :

$$C_{L_n,L_{n+1}} = \log_{10} \left| \left(\frac{L_n}{K(\frac{h_0}{n})^p}\right) \left(\frac{1}{1 - (\frac{n}{n+1})^p}\right) \right| + \mathcal{O}\left(n^{p-q}\right) \tag{4.57}$$

Enfin

$$C_{L_n,L_{n+1}} = C_{L_n,L} + \log_{10} \left(\frac{1}{1 - (\frac{n}{n+1})^p}\right) + \mathcal{O}\left(n^{p-q}\right). \tag{4.58}$$

□

$\frac{1}{1-(\frac{n}{n+1})^p} \leq 2$ si et seulement si $n \leq M_p$, avec $M_p = \frac{1}{2^{\frac{1}{p}}-1}$. Le terme $\log_{10}\left(\frac{1}{1-(\frac{n}{n+1})^p}\right)$ représente alors au plus un bit. Plus l'ordre p de la méthode est élevé, plus la valeur M_p est élevée elle aussi.

Supposons que le nombre total d'itérations effectuées soit inférieur ou égal à $M_p + 1$. Lorsque la zone de convergence est atteinte, *i.e.* lorsque le terme $\mathcal{O}\left(n^{p-q}\right)$ devient négligeable, les bits communs à deux approximations successives, sont aussi communs à la solution exacte, à un près.

Avec la même restriction sur le nombre d'itérations, si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux approximations successives soit non significative, alors les bits significatifs exacts de la dernière approximation sont en commun avec la solution exacte, à un près.

La condition sur le nombre d'itérations effectuées est trop restrictive pour la méthode des trapèzes, d'ordre 2. En effet, dans ce cas, $M_p \approx 2.4$.

Comme la méthode de Simpson nécessite un nombre pair de subdivisions de l'intervalle d'intégration $[a,b]$, une suite d'approximations (I_n) ne peut pas être construite avec un pas de la forme $\frac{b-a}{n}$, mais $\frac{b-a}{2n}$. Dans l'équation (4.58), on retrouve alors, avec $p = 4$, le terme $\log_{10}(\frac{2^p}{2^p-1})$ qui était déjà présent dans le théorème 7 et représente au plus un bit.

Le théorème 13 peut être intéressant pour des méthodes d'ordre relativement élevé telle la méthode de Gauss-Legendre.
Avec la méthode de Gauss-Legendre à 6 points, si l'on effectue moins de 18 itérations, avec le test d'arrêt précédemment décrit, on peut obtenir une approximation dont les bits significatifs exacts sont communs avec la valeur exacte de l'intégrale, à un près.
L'approximation obtenue avec la méthode de Gauss-Legendre à 12 points peut avoir la même propriété si le nombre d'itérations effectuées reste inférieur à 36.
Lors d'une expérimentation numérique décrite en 5.5, le théorème 13 a permis de réduire considérablement le temps de calcul d'une application. En effet, le résultat pouvait être obtenu par la méthode de Gauss-Legendre à 12 points avec 5 subdivisions de l'intervalle d'intégration. Avec la stratégie décrite en 4.3.2, 8 subdivisions auraient été nécessaires.

4.5.2 Généralisation

Les théorèmes présentés ci-après permettent de généraliser les stratégies proposées précédemment. Ils sont fondés sur la comparaison d'itérés d'une suite qui ne sont pas nécessairement successifs.

Lors du calcul d'une suite (I_n) à convergence linéaire, le théorème 2 permet à partir de deux itérés successifs I_n et I_{n+1} de déterminer les premiers chiffres de la limite exacte. Le théorème 14 est fondé sur la comparaison de deux itérés I_n et I_{n+r} d'une suite à convergence linéaire.

Théorème 14 *Considérons une suite (I_n) qui converge linéairement vers I, i.e. qui vérifie $I_n - I = K\alpha^n + o(\alpha^n)$ où $K \in \mathbb{R}$ et $0 < |\alpha| < 1$. Soit $r \in \mathbb{N}^*$, alors*

$$C_{I_n,I_{n+r}} = C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha^r}\right) + o\left(1\right).$$

Preuve De l'erreur sur I_n, on déduit :

$$\frac{I_n + I}{2(I_n - I)} = \frac{I_n}{I_n - I} - \frac{1}{2} = \frac{I_n}{K\alpha^n} + o\left(\frac{1}{\alpha^n}\right) \tag{4.59}$$

De l'erreur sur I_{n+r}, on déduit :

$$I_n - I_{n+r} = K\alpha^n(1-\alpha^r) + o(\alpha^n) \tag{4.60}$$

Donc

$$\frac{I_n + I_{n+r}}{2(I_n - I_{n+r})} = \frac{I_n}{I_n - I_{n+r}} - \frac{1}{2} = \frac{I_n}{K\alpha^n}\,\frac{1}{1-\alpha^r} + o\left(\frac{1}{\alpha^n}\right) \tag{4.61}$$

De la définition 9 et de l'équation (4.59), on déduit :

$$C_{I_n,I} = \log_{10}\left|\frac{I_n}{K\alpha^n}\right| + o(1) \tag{4.62}$$

De même, de la définition 9 et de l'équation (4.61), on déduit :

$$C_{I_n,I_{n+r}} = \log_{10}\left|\frac{I_n}{K\alpha^n}\ \frac{1}{1-\alpha^r}\right| + o\,(1) \tag{4.63}$$

Enfin

$$C_{I_n,I_{n+r}} = C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha^r}\right) + o\,(1) \tag{4.64}$$

□

Lorsque la zone de convergence est atteinte, *i.e.* lorsque $o\,(1) \ll 1$, les chiffres significatifs communs à I_n et I_{n+r} sont aussi communs avec la limite exacte I, à $\log_{10}\left(\frac{1}{1-\alpha^r}\right)$ près.
Si $|\alpha| \leq 2^{-\frac{1}{r}}$, alors $\log_2\left(\frac{1}{1-\alpha^r}\right) \leq 1$. Dans ce cas, si zone de convergence est atteinte, les bits significatifs communs à I_n et I_{n+r} sont aussi communs avec la limite exacte I, à un près.

Un résultat théorique similaire au théorème 14 peut être établi pour les suites à convergence exponentielle.

Théorème 15 *Considérons une suite (I_n) qui converge vers I avec une vitesse exponentielle, i.e. qui vérifie $I_n - I = K\,\alpha^{p^n} + o(\alpha^{p^n})$ où $K \in \mathbb{R}$, $0 < |\alpha| < 1$ et $p > 1$. Soit $r \in \mathbb{N}^*$, alors*

$$C_{I_n,I_{n+r}} = C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha^{p^n(p^r-1)}}\right) + o\,(1)\,.$$

Preuve De l'erreur sur I_n, on déduit :

$$\frac{I_n+I}{2(I_n-I)} = \frac{I_n}{I_n-I} - \frac{1}{2} = \frac{I_n}{K\alpha^{p^n}} + o\left(\frac{1}{\alpha^{p^n}}\right) \tag{4.65}$$

De l'erreur sur I_{n+r}, on déduit :

$$I_n - I_{n+r} = K\,\left(\alpha^{p^n} - \alpha^{p^{n+r}}\right) + o(\alpha^{p^n}) \tag{4.66}$$

Donc

$$\frac{I_n+I_{n+r}}{2(I_n-I_{n+r})} = \frac{I_n}{I_n-I_{n+r}} - \frac{1}{2} = \frac{I_n}{K\,\left(\alpha^{p^n} - \alpha^{p^{n+r}}\right)} + o\left(\frac{1}{\alpha^{p^n}}\right) \tag{4.67}$$

De la définition 9 et de l'équation (4.65), on déduit :

$$C_{I_n,I} = \log_{10}\left|\frac{I_n}{K\alpha^{p^n}}\right| + o(1) \tag{4.68}$$

De même, de la définition 9 et de l'équation (4.67), on déduit :

$$C_{I_n,I_{n+r}} = \log_{10}\left|\frac{I_n}{K\,\alpha^{p^n}\,\left(1-\alpha^{p^n(p^r-1)}\right)}\right| + o(1) \tag{4.69}$$

Enfin

$$C_{I_n,I_{n+r}} = C_{I_n,I} + \log_{10}\left(\frac{1}{1-\alpha^{p^n(p^r-1)}}\right) + o(1) \tag{4.70}$$

□

Lorsque la zone de convergence est atteinte, *i.e.* lorsque $o(1) \ll 1$, les chiffres significatifs communs à I_n et I_{n+r} sont aussi communs avec la limite exacte I, à $\log_{10}\left(\frac{1}{1-\alpha^{p^n(p^r-1)}}\right)$ près. Si $|\alpha| \leq 2^{\frac{1}{p^n(1-p^r)}}$, alors $\log_2\left(\frac{1}{1-\alpha^{p^n(p^r-1)}}\right) \leq 1$. Dans ce cas, si zone de convergence est atteinte, les bits significatifs communs à I_n et I_{n+r} sont aussi communs avec la limite exacte I, à un près. Cette condition sur α est aisément vérifiée. En effet, dans le cas d'une convergence quadratique (*i.e.* pour $p = 2$) si $n = 5$ et $r = 1$, $2^{\frac{1}{p^n(1-p^r)}} > 0.97$. Si r augmente, la condition que α doit vérifier pour que $\log_2\left(\frac{1}{1-\alpha^{p^n(p^r-1)}}\right) \leq 1$ devient de moins en moins stricte.

Supposons que l'on calcule une suite à convergence linéaire telle que $|\alpha| \leq 2^{-\frac{1}{r}}$ ou une suite à convergence exponentielle telle que $|\alpha| \leq 2^{\frac{1}{p^n(1-p^r)}}$. Si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux itérés soit non significative, alors les bits significatifs exacts du dernier itéré sont en commun avec la limite exacte, à un près.

Lors du contrôle dynamique d'une méthode d'approximation, on peut parfois économiser en temps de calcul en accélérant la progression du pas utilisé. Le théorème 16 est une généralisation du théorème 13. Il permet à partir d'approximations calculées avec les pas $\frac{h_0}{n}$ et $\frac{h_0}{n+r}$ d'en déduire les premiers chiffres significatifs de la valeur exacte correspondante.

Théorème 16 *Considérons une méthode numérique qui fournit une approximation $L(h)$ d'ordre p d'une valeur exacte L, i.e. $L(h) - L = Kh^p + \mathcal{O}(h^q)$ avec $1 \leq p < q$, $K \in \mathbb{R}$. Si L_n est l'approximation calculée avec le pas $\frac{h_0}{n}$ et $r \in \mathbb{N}^*$, alors*

$$C_{L_n,L_{n+r}} = C_{L_n,L} + \log_{10}\left(\frac{1}{1-(\frac{n}{n+r})^p}\right) + \mathcal{O}\left(n^{p-q}\right).$$

La preuve de ce théorème est similaire à celle du théorème 13.

Dans le cas où $r = 1$, on retrouve le théorème 13 et dans le cas où $n = r = 2^m$, on retrouve le théorème 7.

Lorsque la zone de convergence est atteinte, *i.e.* lorsque $\mathcal{O}(n^{p-q}) \ll 1$, les chiffres significatifs communs à L_n et L_{n+r} sont aussi communs avec la valeur exacte L, à $\log_{10}\left(\frac{1}{1-(\frac{n}{n+r})^p}\right)$ près. Si $\frac{r}{n} \geq 2^{\frac{1}{p}} - 1$, alors $\log_2\left(\frac{1}{1-(\frac{n}{n+r})^p}\right) \leq 1$. Dans ce cas, si la zone de convergence est atteinte, les bits significatifs communs à L_n et L_{n+r} sont aussi communs avec la valeur exacte L, à un près. Si $r = n$, $\log_2\left(\frac{1}{1-(\frac{n}{n+r})^p}\right)$ représente au plus un bit, quel que soit l'ordre p de la méthode.
Par conséquent, sous réserve que la condition $\frac{r}{n} \geq 2^{\frac{1}{p}} - 1$ soit satisfaite, si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux approximations L_n et L_{n+r} soit non significative, alors les bits significatifs exacts de la dernière approximation sont en commun avec la valeur exacte L, à un près.

Il n'est pas souhaitable d'appliquer cette stratégie de calcul à la méthode des trapèzes ou de Simpson. En effet, la stratégie décrite en 4.3.1, fondée sur des divisions successives du pas d'intégration par 2, est préférable puisqu'elle permet de toujours réutiliser les points calculés précédemment.

Il peut être intéressant d'utiliser le théorème 16 pour le calcul d'une intégrale par la méthode de Gauss-Legendre. Soit I_n l'approximation obtenue par n découpages du domaine d'intégration en sous-intervalles sur lesquels on applique la méthode classique de Gauss-Legendre à ν points. Lorsque le nombre n de découpages augmente, on ne peut généralement pas réutiliser les points calculés précédemment. Lorsqu'on souhaite effectuer un contrôle dynamique du calcul, on peut alors parfois économiser en temps d'exécution en incrémentant de r le nombre de découpages de l'intervalle d'intégration. Cette stratégie a d'ailleurs été utilisée pour une application décrite en 8.9.
Si $n \le M_{\nu,r}$ avec $M_{\nu,r} = \frac{r}{2^{\frac{1}{2\nu}}-1}$, alors $\log_2\left(\frac{1}{1-(\frac{n}{n+r})^{2\nu}}\right) \le 1$. Cette condition est relativement aisément satisfaite. En effet, si $\nu = 6$ et $r = 1$, alors $\lfloor M_{\nu,r} \rfloor = 16$, la borne $M_{\nu,r}$ augmentant si ν ou r augmente. Dans ce cas, d'après le théorème 16, les bits significatifs communs à deux approximations I_n et I_{n+r} dans la zone de convergence sont aussi communs avec la valeur exacte de l'intégrale, à un près.

4.6 Calcul d'une intégrale issue de la cristallographie

L'étude présentée ici est décrite en détail en 8.8. Considérons l'intégrale

$$g(a) = \int_0^{+\infty} f(x)dx, \text{ avec } f(x) = (\exp^x + \exp^{-x})^a - \exp^{ax} - \exp^{-ax} \text{ et } 0 < a < 2.$$

Celle-ci provient d'une étude de l'énergie électronique totale de cristaux [81], où $g(5/3)$ a été évalué à 4.45. Comme il semblait que ce résultat pouvait être amélioré, le problème de l'évaluation de $g(5/3)$ avec la formule suivante a été posé dans [157].

$$g(a) = \sum_{n=0}^{+\infty} \frac{(-a)_n}{n!} \frac{(-1)^n}{2n-a} = \sum_{n=1}^{+\infty} \frac{\Pi_{i=0}^{n-1}(a-i)}{(n!)(2n-a)} - \frac{1}{a}.$$

Dans cette note, il est indiqué que “d'après plusieurs solveurs, $g(5/3) \approx 4.6262911$”.
Nous montrons qu'avec un contrôle dynamique, il est possible d'effectuer, avec la méthode de Romberg, un calcul efficace et fiable de $g(a)$, pour toute valeur de a dans l'intervalle]0,2[.

Plusieurs problèmes numériques se posent pour le calcul de $g(a)$ par la méthode de Romberg :

- tout d'abord, pour les valeurs élevées de x, des cancellations peuvent se produire lors du calcul de $f(x)$,
- de plus, l'intégrale comporte une borne infinie
- enfin, la méthode de Romberg consiste à calculer une suite d'approximations de l'intégrale. Il peut être alors problématique de choisir l'itéré minimisant l'erreur globale qui se compose à la fois de l'erreur de troncature et de l'erreur d'arrondi.

Afin d'éviter les cancellations, on utilise une expression différente de la fonction f pour chaque borne de l'intervalle. Pour combiner efficacité et fiabilité, on utilise l'expression suivante de

$g(a)$:

$$g(a) = \int_0^l f_1(x)dx + \int_l^{+\infty} f_2(x)dx,\ l \in I\!R^{+*},$$

où $f_1(x) = \exp^{ax}\left((1+\exp^{-2x})^a - 1 - \exp^{-2ax}\right)$
$f_2(x) = \exp^{ax} u(x) - \exp^{-ax}$,
$u(x) = \lim_{n\to\infty} u_n(x)$ avec $u_n(x) = \sum_{i=1}^{n-1} \frac{\exp^{-2ix}}{i!} \Pi_{j=0}^{i-1}(a-j)$.

Pour le calcul de $\int_l^{+\infty} f_2(x)dx$, $u(x)$ doit être approché par un itéré $u_n(x)$. De plus, cette intégrale impropre doit être approchée par une intégrale sur un intervalle fini, telle $s_k = \int_l^{kl} f_2(x)dx$ ($k \in I\!N$, $k \geq 2$). Il faut alors choisir un itéré de la suite (s_k) pour lequel l'indice k est suffisamment élevé. Enfin la méthode de Romberg consiste à calculer une suite d'approximations. Par conséquent, le calcul de $g(a)$ implique plusieurs suites, dont il peut être difficile de choisir l'itéré optimal.

Nous présentons en 8.8 une stratégie dynamique pour calculer avec la méthode de Romberg une approximation de $g(a)$ qui minimise l'erreur globale. De plus, en combinant plusieurs résultats théoriques décrits dans les chapitres 2 et 4, nous montrons que les chiffres décimaux significatifs exacts de l'approximation obtenue, *i.e.* ceux qui ne sont pas entachés d'erreur d'arrondi, sont en commun avec la valeur mathématique, à $\delta = \log_{10}\left(2/(1-\exp^{-l\min(a,2-a)})\right)$ près.

Pour différentes valeurs de a, l'intégrale $g(a)$ a été calculée en ASD en double précision avec la stratégie décrite en 8.8. Pour chaque approximation $G(a)$ obtenue, ses chiffres significatifs exacts, *i.e.* ses chiffres significatifs qui ne sont pas entachés d'erreur d'arrondi, ont été comparés avec les seize premiers chiffres de la valeur exacte $g(a)$. Ces seize chiffres exacts ont été obtenus en calculant le développement en série de $g(a)$ avec la bibliothèque MPFI [161], qui implante l'arithmétique d'intervalles en multiprécision. Cette bibliothèque fournit des bornes garanties de résultats obtenus avec une précision arbitraire. Nous décrivons ci-après une stratégie fondée sur les propriétés du développement en série de $g(a)$, qui nous permet de calculer ses chiffres exacts.

L'intégrale $g(a)$ peut être exprimée par un développement en série.
En effet, $g(a) = \int_0^\infty \left(\exp^{ax}\sum_{i=1}^\infty \frac{\exp^{-2ix}}{i!}\Pi_{j=0}^{i-1}(a-j) - \exp^{-ax}\right)dx$.
Alors $g(a) = \sum_{i=1}^\infty \frac{\Pi_{j=0}^{i-1}(a-j)}{i!}\int_0^\infty \exp^{(a-2i)x}\,dx - \int_0^\infty \exp^{-ax}\,dx$.
Par conséquent, $g(a) = \lim_{n\to\infty} w_n$,
avec $w_n = \sum_{i=1}^n v_i - \frac{1}{a}$ et $\forall i \geq 1,\ \ v_i = \frac{\Pi_{j=0}^{i-1}(a-j)}{i!\,(2i-a)}$

L'erreur due à l'approximation de $g(a)$ par un itéré w_n vérifie l'équation (4.71).

$$\forall n \geq 2,\ |g(a) - w_n| \leq |v_{n+1}|. \tag{4.71}$$

L'équation (4.71) a été établie en 8.8 grâce au fait que $\sum_{i=2}^n v_i$ est une série alternée.

Par conséquent, pour tout $\varepsilon > 0$, si la suite (w_n) est calculée jusqu'à ce que $|\frac{v_{n+1}}{w_n}| < \varepsilon$, alors $|\frac{g(a)-w_n}{w_n}| < \varepsilon$: l'erreur relative sur l'approximation w_n obtenue est inférieure à ε. Afin d'obtenir les seize premiers chiffres exacts de $g(a)$, les calculs ont été effectués avec MPFI jusqu'à ce que w_n soit représenté par un intervalle avec une erreur relative inférieure à 10^{-16} et $|\frac{v_{n+1}}{w_n}| < 10^{-16}$. La précision demandée était de cent bits, ce qui correspond à trente chiffres décimaux.

Le tableau 4.4 présente, pour différentes valeurs de a, les approximations $G(a)$ obtenues en ASD et les seize premiers chiffres de la valeur exacte $g(a)$. Seuls les chiffres significatifs exacts de $G(a)$ sont indiqués : les autres chiffres significatifs sont entachés d'erreur d'arrondi. Les chiffres significatifs exacts de $G(a)$ en commun avec la valeur exacte $g(a)$ sont soulignés. Chaque approximation $G(a)$ a été évaluée en calculant avec la méthode de Romberg des intégrales sur un intervalle de longueur $l = 5$. D'après 8.8, le nombre de chiffres décimaux significatifs exacts de $G(a)$ qui ne sont pas en commun avec $g(a)$ est approché par $\delta = \log_{10}\left(2/(1-\exp^{-l\min(a,2-a)})\right)$.

$G(a)$ est une somme d'intégrales calculées sur un intervalle de longueur l par la méthode de Romberg. Dans le tableau 4.4, est reporté le nombre d'intégrales calculées pour chaque approximation $G(a)$. Le nombre d'itérations effectuées avec la méthode de Romberg dépend de l'intégrale calculée. Pour chaque approximation $G(a)$, le minimum et le maximum du nombre d'itérations nécessaires sont indiqués dans le tableau 4.4. Chaque approximation de $u(x)$ est effectuée grâce à son développement en série. Le nombre de termes nécessaires dépend alors de x. Le minimum et le maximum de termes calculés sont aussi indiqués dans le tableau 4.4.

a	$\delta \approx$		$g(a)$	nombre de termes		
				intégrales	Romberg	série
10^{-4}	3.6	exact:	-9.999999958873644E+003			
		ASD:	-9.99999992492E+003	39003	[3,9]	[2,5]
10^{-3}	2.6	exact:	-9.999995884658167E+002			
		ASD:	-9.999995884276E+002	4799	[4,9]	[2,5]
0.5	0.34	exact:	-1.694426169587958E+000			
		ASD:	-1.69442616958795E+000	15	[8,10]	[2,5]
0.9999	0.30	exact:	-3.141471326607299E-004			
		ASD:	-3.1414713265E-004	7	8	[2,4]
1.0001	0.30	exact:	3.141714042514768E-004			
		ASD:	3.1417140425E-004	7	8	[2,4]
$\frac{5}{3}$	0.39	exact:	4.626291111983995E+000			
		ASD:	4.626291111983E+000	19	[7,9]	[2,5]
1.99	1.6	exact:	1.989868510333895E+002			
		ASD:	1.9898685103326E+002	563	[4,8]	[2,4]
1.9999	3.6	exact:	1.999899986776092E+004			
		ASD:	1.99989997358E+004	37676	[3,7]	[2,4]

TAB. 4.4 – *Approximations de $g(a)$ obtenues en ASD*

Pour chaque approximation $G(a)$, le nombre d'intégrales calculées, le nombre d'itérations par la méthode de Romberg et le nombre de termes dans le développement en série de $u(x)$ sont déterminés dynamiquement grâce aux tests d'arrêt décrits en 8.8. Le nombre d'intégrales calculées dépend de la valeur de a. Par exemple, l'évaluation de $g(a)$ nécessite le calcul de 39003 intégrales si $a = 0.0001$ et de seulement 7 intégrales si $a = 0.9999$ ou si $a = 1.0001$. Le nombre d'itérations par la méthode de Romberg varie pour chaque approximation $G(a)$, à l'exception des cas où $a = 0.9999$ et $a = 1.0001$. Le nombre de termes dans le développement en série varie également. Par exemple, si $a = 0.0001$, de 3 à 9 itérations par la méthode de Romberg sont nécessaires et de 2 à 5 termes dans le développement en série sont calculés pour les différentes approximations de $u(x)$.

On remarque que le nombre de chiffres significatifs exacts de $G(a)$ varie de 11 à 15, selon la valeur de a. Par exemple, $G(0.5)$ possède 15 chiffres significatifs exacts et est donc de qualité numérique optimale pour un résultat obtenu en double précision. L'intégrale $g(a)$ converge pour $0 < a < 2$: $\lim_{a\to 0} g(a) = -\infty$ et $\lim_{a\to 2} g(a) = +\infty$. Lorsque a tend vers 0 ou 2, le nombre de chiffres significatifs exacts de $G(a)$ décroît légèrement : $G(a)$ a 12 chiffres significatifs exacts si $a = 10^{-4}$ ou $a = 1.9999$. Si $a = 1$, $\forall x \in [0, +\infty[$, $f(x) = 0$. Si a est proche de 1, il peut se produire des cancellations lors du calcul de $f(x)$. En effet, le nombre de chiffres significatifs exacts de $G(a)$ décroît lorsque a tend vers 1 : $G(a)$ a 11 chiffres significatifs exacts si $a = 0.9999$ ou si $a = 1.0001$.

Les chiffres significatifs exacts de l'approximation $G(a)$ en commun avec $g(a)$ s'identifient aisément ; ils sont soulignés dans le tableau 4.4. Par exemple, si $a = 1.9999$, parmi les 12 chiffres significatifs exacts de $G(a)$, les 4 derniers ne sont pas en commun avec $g(a)$. On remarque que, pour chaque approximation de $G(a)$ reportée dans le tableau 4.4, ses chiffres significatifs exacts sont en commun avec $g(a)$, à $\lceil \delta \rceil$ près.

L'évaluation de $g(5/3)$ était un problème posé dans [157]. Par intégration numérique, W. Harrison [81] avait obtenu pour $g(5/3)$ la valeur 4.45. Ce résultat peut être comparé avec l'approximation $G(5/3)$ indiquée dans le tableau 4.4. L'approximation $G(5/3)$ a 13 chiffres significatifs exacts, qui sont en commun avec la valeur mathématique $g(5/3)$, à un près.

4.7 Conclusion et perspectives

Les résultats théoriques présentés dans ce chapitre ont été établis à partir du développement de l'erreur de troncature commise. Fondés sur le calcul d'une suite d'approximations, ils permettent, à partir de deux approximations, de déterminer les premiers chiffres de la limite exacte. Les approximations obtenues sur ordinateur sont affectées d'une erreur globale formée de l'erreur de troncature et de l'erreur d'arrondi. Si les calculs sont effectués jusqu'à ce que la différence entre deux approximations soit non significative, alors l'erreur globale sur le résultat est minimale. De plus, si la suite converge suffisamment vite, les bits significatifs du résultat qui ne sont pas entachés d'erreur d'arrondi sont ceux de la limite exacte, à un près. Certains algorithmes numériques nécessitent des approximations imbriquées, par exemple lors du calcul d'une suite dont les itérés sont eux-mêmes des limites de suites. Des stratégies ont alors été proposées afin d'obtenir un résultat de précision optimale. Les théorèmes présentés dans les chapitres 2 et 4 peuvent être combinés pour permettre de déterminer quels chiffres significatifs du résultat obtenu en ASD sont en commun avec la valeur exacte correspondante. Il est ainsi possible de garantir les premiers chiffres significatifs de l'approximation d'une intégrale sur un intervalle infini ou, comme nous le montrerons dans le chapitre suivant, d'une intégrale multiple.

Les stratégies proposées dans ce chapitre sont fondées sur le calcul d'une suite d'approximations. Chaque itéré correspond alors à un pas de discrétisation constant. Dans le cas de l'approximation d'une intégrale, l'intervalle d'intégration est divisé en sous-intervalles de même longueur. Or utiliser une méthode d'approximation avec un pas adaptatif permet d'économiser parfois en temps de calcul. Avec une méthode de quadrature, cela consiste à augmenter la valeur du pas lorsque la fonction à intégrer varie peu. Il serait intéressant d'établir en ASD une stratégie avec un pas adaptatif et surtout de garantir le résultat obtenu.

Les travaux présentés dans ce chapitre peuvent aussi être poursuivis dans d'autres directions. Un théorème a été établi pour le contrôle dynamique de la méthode de Gauss-Legendre, la méthode de quadrature de Gauss la plus couramment utilisée. Des résultats théoriques similaires pourraient être proposés pour les autres méthodes de quadrature de Gauss, qui permettent le calcul d'intégrales sur un domaine fini (telle la méthode de Gauss-Chebyshev ou de Gauss-Jacobi) ou infini (telle la méthode de Gauss-Laguerre ou de Gauss-Hermite).

Les résultats théoriques obtenus pour la méthode de Romberg pourraient être étendus à d'autres méthodes d'extrapolation, qui permettent d'accélérer la convergence de suites. Le stage de DEA de R. Adout [3], que j'ai co-encadré avec J.-M. Chesneaux, portait sur la validation numérique des méthodes d'extrapolation. R. Adout a alors étudié d'une part la méthode d'Aitken et sa généralisation, l'ε-algorithme, d'autre part la méthode de Romberg sa généralisation, le procédé de Richardson [25]. Ces méthodes ont été utilisées pour accélérer la convergence de suites relativement lentes générées par le calcul d'intégrales de fonctions non dérivables ou par la résolution d'équations différentielles. Une étude comparative de la qualité numérique de ces différentes méthodes a pu être effectuée grâce à l'ASD. Cependant le théorème 12, établi à partir du développement de l'erreur de troncature de Romberg, n'a pas encore d'équivalent pour les autres méthodes d'extrapolation.

Les travaux présentés dans ce chapitre concernent le contrôle dynamique du calcul de suites de scalaires. Une perspective importante est leur extension aux suites de vecteurs. Bien que plusieurs études aient déjà été consacrées au contrôle dynamique de méthodes itératives de résolution de systèmes linéaires, beaucoup reste à faire dans ce domaine.
La thèse de M. Montagnac [142], soutenue en 1999, a porté sur le contrôle dynamique des méthodes BICGSTAB, CGS et GMRES [164]. Les stratégies qu'il a proposées utilisent l'ASD et sont fondées sur des redémarrages dynamiques de ces méthodes.
La thèse de R. Adout, que je co-encadre avec J.-M. Chesneaux, porte notamment sur l'accélération de la convergence, par des méthodes d'extrapolation, de la méthode GMRES redémarrée. Une méthode hybride fondée sur l'algorithme VEA (Vector Epsilon Algorithm) [25] a été proposée [4]. L'ASD est alors utilisée pour s'assurer de la qualité numérique de cette méthode.
Les méthodes itératives de résolution de systèmes linéaires utilisent divers paramètres. Dans le cas de la méthode GMRES redémarrée, il s'agit de la dimension du sous-espace de Krylov sur lequel s'effectuent les projections et du nombre de redémarrages. Un travail important

reste à faire afin de déterminer comment effectuer un contrôle dynamique des paramètres inhérents à chaque méthode.

Chapitre 5

Contrôle dynamique du calcul d'intégrales multiples

5.1 Diverses approches pour le calcul d'intégrales multiples

Les intégrales multiples apparaissent dans de nombreux domaines d'application tels la mécanique ou la finance. Elles peuvent être évaluées en utilisant diverses approches que nous présentons ci-après.

Considérons l'intégrale de dimension N

$$I[f] = \int_{\Omega} w(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \tag{5.1}$$

où $\Omega \subset I\!R^N$ et $\forall \mathbf{x} \in \Omega$, $w(\mathbf{x}) \geq 0$.

Cette intégrale peut être approchée par une somme pondérée d'évaluations de la fonction f :

$$Q[f] = \sum_{j=1}^{\nu} a_j f(\mathbf{x_j}), \tag{5.2}$$

où $a_j \in I\!R$ et $\mathbf{x_j} \in \Omega$.

L'approximation Q est appelée *formule de quadrature* si $N = 1$ et *formule de cubature* si $N \geq 2$.

L'approximation (5.2) étant supposée exacte si la fonction f est constante,

$$I[1] = \int_{\Omega} w(\mathbf{x}) d\mathbf{x} = \sum_{j=1}^{\nu} a_j = Q[1]. \tag{5.3}$$

On distingue principalement deux classes de formules de cubature : les méthodes fondées sur les polynômes et les méthodes de Monte Carlo. Le choix de ces méthodes dépend de la dimension de l'intégrale. En 1984, P.J. Davis et P. Rabinowitz proposaient une classification des intégrales selon leur dimension [60]. En 2002, R. Cools considérait cette classification comme étant toujours d'actualité [49]. Pour les intégrales de dimension inférieure à 8, on utilise des méthodes fondées sur des formules exactes pour certains polynômes. Les méthodes de Monte Carlo ou de quasi-Monte Carlo [28, 143, 148] deviennent compétitives pour les intégrales de dimension 8 à 15 environ et sont particulièrement indiquées pour les intégrales de dimension supérieure à 15. Ce chapitre n'abordant pas le calcul d'intégrales de dimension très élevée, nous décrivons ci-après les méthodes fondées sur les polynômes.

Un critère permettant de classifier les méthodes de cubature fondées sur les polynômes est leur degré. Une formule de cubature est de degré (polynomial algébrique) d si elle est exacte pour tout polynôme de degré au plus d.
Une intégrale multiple peut être calculée par le principe des "intégrales itérées" [167]. Elle est alors décomposée en différentes intégrales monodimensionnelles qui peuvent être évaluées par une méthode de quadrature. Le principe des "intégrales itérées" équivaut à effectuer un produit cartésien de plusieurs régions monodimensionnelles [77]. En effet, considérons les formules de quadrature suivantes, respectivement de degré d_1 et d_2, pour l'approximation d'intégrales sur les régions monodimensionnelles R_1 et R_2.

$$\int_{R_1} w_1(x) f_1(x) dx \approx \sum_{i=1}^{\nu_1} a_i f_1(x_i), \tag{5.4}$$

où $\forall i = 1,...,\nu_1$, $a_i \in I\!R$ et $x_i \in R_1$.

$$\int_{R_2} w_2(y) f_2(y) dy \approx \sum_{j=1}^{\nu_2} a_j f_2(y_j), \tag{5.5}$$

où $\forall j = 1,...,\nu_2$, $a_j \in I\!R$ et $y_j \in R_2$.

À partir de ces deux formules de quadrature, on peut obtenir une formule de cubature pour l'approximation de

$$I = \int_{R_1 \times R_2} w_1(x) w_2(y) f(x,y) dx dy. \tag{5.6}$$

L'intégrale I peut s'écrire

$$I = \int_{R_1} w_1(x) f_1(x) dx, \tag{5.7}$$

où $\forall x \in R_1$, $f_1(x) = \int_{R_2} w_2(y) f(x,y) dy$.

$$I \approx \sum_{i=1}^{\nu_1} a_i \int_{R_2} w_2(y) f(x_i,y) dy. \tag{5.8}$$

Donc

$$I \approx \sum_{i=1}^{\nu_1} \sum_{j=1}^{\nu_2} a_i a_j f(x_i,y_j). \tag{5.9}$$

La formule de cubature (5.9) est alors de degré $d = \min(d_1,d_2)$ [131].

Utiliser directement une formule de cubature telle (5.9) ou calculer plusieurs intégrales monodimensionnelles grâce au principe des "intégrales itérées" peut donc être mathématiquement équivalent, mais conduire à une implémentation différente.

Soit $\mathbb{P}_d^N$ le sous-espace des polynômes de N variables de degré au plus d. Par définition, pour une formule de cubature de degré d, si les polynômes f_i constituent une base de $\mathbb{P}_d^N$,

$$Q[f_i] = I[f_i] \ (i = 1,2,..., \dim \mathbb{P}_d^N), \tag{5.10}$$

c'est-à-dire

$$\sum_{j=1}^{\nu} a_j f_i(\mathbf{x_j}) = I[f_i] \ (i = 1,2,..., \dim \mathbb{P}_d^N). \tag{5.11}$$

Les poids a_j et les points $\mathbf{x_j}$ d'une formule de cubature peuvent donc être déterminés grâce au système d'équations non linéaires (5.11).

On sait actuellement que, pour un degré donné, le nombre minimal de points dépend du domaine d'intégration et de la fonction de poids w [49]. Néanmoins on ne connaît toujours pas le nombre minimal de points pour des domaines simples tels un carré ou un triangle avec une fonction de poids constante.

D'après [55], une variété significative de formules de cubature ont été établies pour seulement quatre domaines bidimensionnels : le carré, le cercle, le triangle unitaire et le plan. Grâce aux transformations affines qui n'affectent pas leur degré, ces formules de cubature peuvent être adaptées à d'autres domaines bidimensionnels tels le parallélogramme ou l'ellipse.

Cubpack est une bibliothèque permettant le calcul d'intégrales multiples. Une première version en Fortran 77 a été écrite en 1992 [51]. Puis une version en C++, nommée Cubpack++, a été développée [53]. La bibliothèque Cubpack++ permet le calcul d'intégrales sur des domaines bidimensionnels qui peuvent être définis à partir des quatre domaines de base cités précédemment. Les formules de cubature implantées sont de degré 13, à 36 ou 37 points selon le domaine de base considéré. Une stratégie adaptative est utilisée. Une nouvelle version de la bibliothèque Cubpack a été écrite en Fortran 95 [52]. Celle-ci permet le calcul d'intégrales de dimension plus élevée. Elle comprend en outre une procédure d'extrapolation utilisant l'ε-algorithme pour les domaines d'intégration non triangulaires. La bibliothèque Cubpack fournit toujours une estimation de l'erreur de troncature commise.

Les algorithmes développés en arithmétique d'intervalles pour le calcul d'intégrales multiples prennent en compte à la fois l'erreur de troncature et l'erreur d'arrondi générées. Les méthodes de cubature qui ont été implantées sont généralement fondées sur le produit de formules de quadrature. Les stratégies qui ont été établies sont adaptatives et utilisent, par exemple, la méthode de Romberg [169], la méthode de Clenshaw-Curtis [35], ou la méthode de Gauss [35, 83, 116, 117, 185]. Deux bibliothèques ont été développées en arithmétique d'intervalles pour le calcul d'intégrales de dimension 1 ou 2 : VANI (Validated Adaptive Numerical Integration) [35] écrite en Pascal-XSC, puis CLAVIS (CLAsses for Verified Integration over Singularities) [185] écrite en C-XSC. Cette dernière bibliothèque est accessible sur internet, sur le site qui présente les langages XSC[1].

1. http://www.xsc.de

Les stratégies de calcul d'intégrales multiples décrites dans la suite de ce chapitre sont fondées sur des subdivisions du domaine d'intégration. En contrôlant dynamiquement le nombre de subdivisions effectuées, on peut obtenir une approximation pour laquelle l'erreur globale, formée de l'erreur de troncature et de l'erreur d'arrondi, est minimale. Nous montrons alors comment déterminer les chiffres significatifs du résultat obtenu qui sont en commun avec la valeur exacte de l'intégrale. Ces stratégies ont été implantées dans l'atelier ACIM (Atelier de Calcul d'Intégrales Multiples). Celui-ci permet le calcul en ASD (Arithmétique Stochastique Discrète) d'intégrales de dimension 1 à 3.

5.2 Approximation par le principe des "intégrales itérées"

Le principe des "intégrales itérées" [167] consiste à calculer une intégrale multiple en la décomposant en plusieurs intégrales monodimensionnelles. L'intégrale multiple peut alors être évaluée en utilisant une méthode de quadrature. Les stratégies de calcul décrites ci-après sont fondées sur les résultats présentés en 4.3. Elles permettent de calculer les différentes intégrales monodimensionnelles avec un pas d'intégration optimal. De plus, on peut déterminer quels chiffres de l'approximation obtenue sont en commun avec la valeur exacte de l'intégrale. Dans ce qui suit, on désigne par des lettres majuscules les valeurs calculées en ASD et par des lettres minuscules les valeurs exactes correspondantes.

5.2.1 Calcul d'intégrales bidimensionnelles

Étant donnée une fonction f continue sur un domaine borné D de dimension 2, on cherche à calculer l'intégrale

$$s = \int_D f(x,y)dxdy = \int_a^b \int_{y_1(x)}^{y_2(x)} f(x,y)dxdy. \tag{5.12}$$

Celle-ci peut s'écrire sous la forme de deux intégrales simples :
$s = \int_a^b g(x)dx$, avec $\forall x \in [a,b],\ \ g(x) = \int_{y_1(x)}^{y_2(x)} f(x,y)dy$.

L'intégrale s peut donc être évaluée par une méthode de quadrature telle la méthode des trapèzes, de Simpson, de Romberg ou de Gauss-Legendre.

L'approximation de s nécessite l'évaluation de l'intégrale $g(x_i)$ pour différents points x_i de l'intervalle $[a,b]$. Chaque intégrale $g(x_i)$ peut être évaluée en utilisant l'une des stratégies de calcul décrites en 4.3. Soit $G_m(x_i)$ l'approximation de $g(x_i)$ calculée :

- en utilisant la méthode des trapèzes ou de Simpson avec le pas $\frac{y_2(x_i)-y_1(x_i)}{2^m}$,
- en partitionnant l'intervalle d'intégration $[y_1(x_i),y_2(x_i)]$ en 2^m sous-intervalles sur lesquels on applique la méthode de Gauss-Legendre
- ou en effectuant m itérations de la méthode de Romberg.

La suite $(G_m(x_i))$ est calculée jusqu'à ce que la différence entre deux itérés successifs soit non significative. Cette condition n'est pas nécessairement vérifiée à la même itération pour chaque valeur x_i. Soit m_i l'itération pour laquelle la différence $G_{m_i-1}(x_i) - G_{m_i}(x_i)$ est non significative. D'après les résultats théoriques décrits en 4.3, les bits significatifs exacts de $G_{m_i}(x_i)$ sont ceux de $g(x_i)$ à δ près, avec

$$\delta = \log_2\left(\frac{2^p}{2^p-1}\right), \tag{5.13}$$

p désignant l'ordre de la méthode de quadrature ($p = 2$ pour la méthode des trapèzes, $p = 4$ pour la méthode de Simpson, $p = 2\nu$ pour la méthode de Gauss-Legendre à ν points) et $\delta = 0$ pour la méthode de Romberg. δ représente au plus un bit.

Soit S_n l'approximation de s calculée avec ces valeurs $G_{m_i}(x_i)$ en utilisant l'une des stratégies de calcul décrites en 4.3. L'approximation S_n peut donc être obtenue :
- en utilisant la méthode des trapèzes ou de Simpson avec le pas $\frac{b-a}{2^n}$,
- en partitionnant l'intervalle d'intégration en 2^n sous-intervalles sur lesquels on applique la méthode de Gauss-Legendre
- ou en effectuant n itérations de la méthode de Romberg.

S_n est une combinaison linéaire des approximations $G_{m_i}(x_i)$: $S_n = \phi(\{G_{m_i}(x_i)\})$, où la fonction ϕ dépend de la méthode d'intégration choisie.
On note s_n la valeur obtenue en appliquant la même méthode de quadrature aux valeurs exactes $g(x_i)$: $s_n = \phi(\{g(x_i)\})$.
Comme S_n est une combinaison linéaire des approximations $G_{m_i}(x_i)$ dont les bits significatifs exacts sont en commun avec la valeur mathématique $g(x_i)$ à δ près, d'après le théorème 1, les bits significatifs exacts de S_n sont ceux de s_n à δ près.
D'après les résultats théoriques décrits en 4.3, lorsque la zone de convergence de la suite (s_n) est atteinte, les bits significatifs communs à s_{n-1} et s_n sont aussi communs à s, à δ près.

Lorsque la différence entre deux approximations successives S_{n-1} et S_n est non significative, tous les bits significatifs exacts de S_n sont en commun avec S_{n-1}. Comme ils sont aussi en commun avec s_n et s_{n-1} à δ près, ils sont alors en commun avec s à 2δ près.
La figure 5.1 représente les bits significatifs de S_{n-1} et S_n lorsque leur différence est non significative.

Quelle que soit la méthode de quadrature choisie, 2δ représente au plus un bit. Les bits significatifs exacts de l'approximation S_n obtenue sont donc en commun avec s, la valeur mathématique de l'intégrale, à un près.

5.2.2 Calcul d'intégrales tridimensionnelles

Étant donnée une fonction f continue sur un domaine borné D de dimension 3, on cherche à calculer l'intégrale

$$s = \int_D f(x,y,z)\,dx\,dy\,dz = \int_a^b \int_{y_1(x)}^{y_2(x)} \int_{z_1(x,y)}^{z_2(x,y)} f(x,y,z)\,dx\,dy\,dz. \tag{5.14}$$

Celle-ci peut s'écrire sous la forme suivante :
$s = \int_a^b h(x)dx$, où $\forall x \in [a,b]$, $h(x) = \int_{y_1(x)}^{y_2(x)} \int_{z_1(x,y)}^{z_2(x,y)} f(x,y,z)\,dy\,dz$.

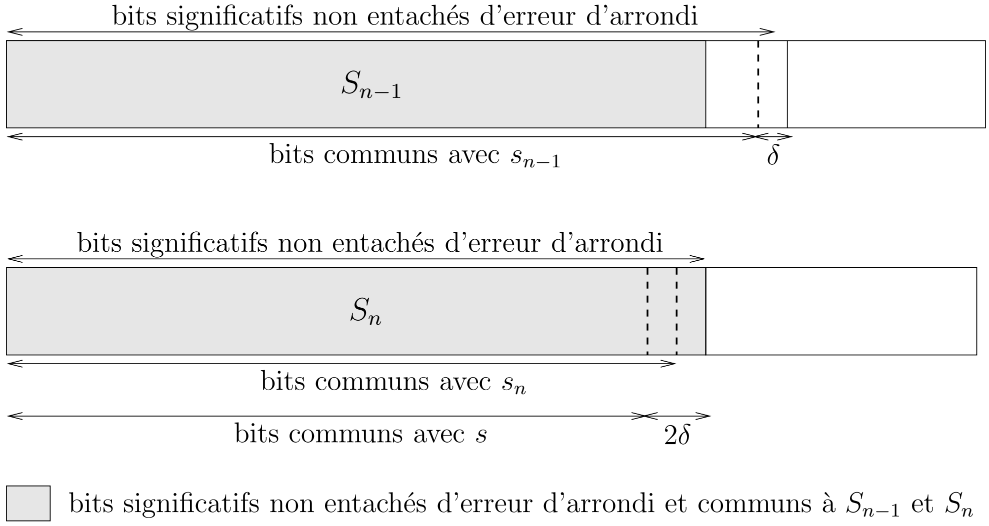

FIG. 5.1 – *Bits significatifs de S_{n-1} et S_n lorsque $S_{n-1} - S_n = @.0$*

L'approximation de s nécessite l'évaluation d'intégrales doubles $h(x_i)$ où $x_i \in [a,b]$.
Soit $H_{m_i}(x_i)$ l'approximation de $h(x_i)$ obtenue en utilisant la stratégie de calcul décrite précédemment. Les bits significatifs exacts de $H_{m_i}(x_i)$ sont donc ceux de $h(x_i)$, à 2δ près.
Soit S_n l'approximation de s calculée avec ces valeurs $H_{m_i}(x_i)$ en utilisant l'une des stratégies de calcul décrites en 4.3. L'approximation S_n peut donc être obtenue :

- en utilisant la méthode des trapèzes ou de Simpson avec le pas $\frac{b-a}{2^n}$,
- en partitionnant l'intervalle d'intégration en 2^n sous-intervalles sur lesquels on applique la méthode de Gauss-Legendre
- ou en effectuant n itérations de la méthode de Romberg.

S_n est une combinaison linéaire des approximations $H_{m_i}(x_i)$: $S_n = \phi(\{H_{m_i}(x_i)\})$, où la fonction ϕ dépend de la méthode d'intégration choisie.
On note s_n la valeur obtenue en appliquant la même méthode de quadrature aux valeurs exactes $h(x_i)$: $s_n = \phi(\{h(x_i)\})$.
Comme S_n est une combinaison linéaire des approximations $H_{m_i}(x_i)$ dont les bits significatifs exacts sont en commun avec la valeur mathématique $h(x_i)$ à 2δ près, d'après le théorème 1, les bits significatifs exacts de S_n sont ceux de s_n, à 2δ près.
D'après les résultats théoriques décrits en 4.3, lorsque la zone de convergence de la suite (s_n) est atteinte, les bits significatifs communs à s_{n-1} et s_n sont aussi communs à s, à δ près.

Lorsque la différence entre deux approximations successives S_{n-1} et S_n est non significative, tous les bits significatifs exacts de S_n sont en commun avec S_{n-1}. Comme ils sont aussi en commun avec s_n et s_{n-1} à 2δ près, ils sont alors en commun avec s à 3δ près.
La figure 5.2 représente les bits significatifs de S_{n-1} et S_n lorsque leur différence est non significative.

Si l'on choisit la méthode des trapèzes, $3\delta = 3\log_2\left(\frac{4}{3}\right) \approx 1.2$. Dans ce cas, les bits significatifs exacts de l'approximation S_n obtenue sont donc en commun avec s, la valeur mathématique de l'intégrale, à deux près.

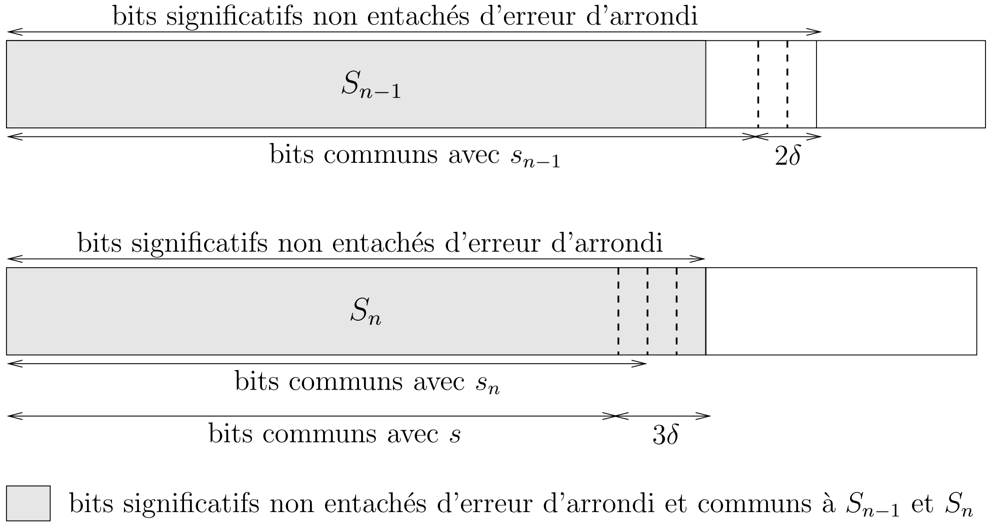

FIG. 5.2 – *Bits significatifs de S_{n-1} et S_n lorsque $S_{n-1} - S_n = @.0$*

Avec les autres méthodes de quadrature, 3δ représente au plus un bit. Les bits significatifs exacts de S_n sont alors en commun avec s, à un près.

5.2.3 Calcul d'intégrales de dimension N

Les stratégies de calcul décrites pour les intégrales bidimensionnelles et tridimensionnelles peuvent être généralisées aux intégrales sur un domaine borné de dimension N. On obtient alors une approximation dont les bits significatifs exacts sont en commun avec la valeur mathématique de l'intégrale à $N\delta$ près.

Avec la méthode de Romberg, $\delta = 0$.

Avec les autres méthodes de quadrature étudiées, $N\delta = \log_2\left(\left(\frac{2^p}{2^p-1}\right)^N\right)$, où p désigne l'ordre de la méthode. On se propose de calculer M_p, la valeur maximale de N telle que $\log_2\left(\left(\frac{2^p}{2^p-1}\right)^N\right) < 1$.

Avec la méthode des trapèzes, $p = 2$ et $M_p = 2$. Cependant si $N \leq 8$, $\log_{10}\left(\left(\frac{2^p}{2^p-1}\right)^N\right) < 1$. Si l'on choisit la méthode des trapèzes, on peut donc obtenir une approximation d'une intégrale de dimension 1 à 8 dont les chiffres décimaux significatifs exacts sont en commun avec la valeur mathématique, à un près.

Avec la méthode de Simpson, $p = 4$ et $M_p = 35$. Si l'on choisit la méthode de Simpson, on peut donc obtenir une approximation d'une intégrale de dimension 1 à 35 dont les bits significatifs exacts sont en commun avec la valeur mathématique, à un près.

La valeur M_p augmente fortement à mesure que l'ordre p augmente. Avec la méthode de Gauss-Legendre à 6 points, $p = 12$ et $M_p = 2838$.

5.3 Approximation par méthodes de cubature

Les méthodes de cubature permettent le calcul d'une intégrale multiple par une formule adaptée au domaine d'intégration. Dans cette partie, nous montrons comment effectuer un contrôle dynamique des méthodes de cubature sur l'hypercube $[0,1]^N$. La stratégie de calcul décrite ci-après sera ensuite généralisée au calcul d'intégrales sur un hyperrectangle de dimension N.

Tout d'abord, nous allons rappeler quelques propriétés des formules de cubature pour l'approximation d'intégrales sur l'hypercube $[0,1]^N$. Ces propriétés ont été énoncées dans diverses publications, notamment [130, 175] et [131] pour le cas bidimensionnel.

Soit

$$I = \int_{[0,1]^N} f(\mathbf{x}) d\mathbf{x}. \tag{5.15}$$

L'intégrale I peut être approchée par une formule de cubature Q à ν points, conforme à l'équation (5.2). La formule de cubature Q étant supposée exacte si la fonction f est constante, les poids a_j vérifient :

$$\sum_{j=1}^{\nu} a_j = 1. \tag{5.16}$$

En divisant l'hypercube $[0,1]^N$ en m^N hypercubes de côté $\frac{1}{m}$ et en appliquant sur chacun la formule de cubature Q avec la translation adéquate, on obtient la formule $Q^{(m)}$ nommée, dans par exemple [130, 131], "m^N-copy rule" :

$$Q^{(m)}[f] = \frac{1}{m^N} \sum_{\mathbf{k} \in \{0,1,\ldots,m-1\}^N} \sum_{j=1}^{\nu} a_j f\left(\frac{\mathbf{x_j} + \mathbf{k}}{m}\right). \tag{5.17}$$

Le théorème 17 présente le développement d'Euler-Maclaurin de $Q^{(m)}f - I$. Il a été établi dans [130], à partir de divers résultats décrits dans [18, 129, 133].

Théorème 17 *Soient I et $Q^{(m)}[f]$ définis par les équations (5.15) et (5.17).*
Supposons que toutes les dérivées partielles de f d'ordre total au plus p soient intégrables sur $[0,1]^N$, alors

$$Q^{(m)}[f] - I = \sum_{s=1}^{l-1} \frac{B_s}{m^s} + \mathcal{O}\left(\frac{1}{m^l}\right), \quad N \leq l \leq p$$

où les coefficients B_s dépendent de Q et de f, mais pas de m.

Les coefficients B_s vérifient l'équation suivante.

$$B_s = \sum_{|\mathbf{s}|=s} c_{\mathbf{s}}(Q) \int_{[0,1]^N} f^{(\mathbf{s})}(\mathbf{x}) d\mathbf{x} \tag{5.18}$$

avec

$$|\mathbf{s}| = |(s_1,s_2,...,s_N)| = \sum_{i=1}^{N} s_i. \tag{5.19}$$

Les coefficients $c_{\mathbf{s}}(Q)$ sont obtenus en appliquant la formule Q à un produit de polynômes de Bernoulli $B_q(x)$. En effet,

$$c_{\mathbf{s}}(Q) = Q[\phi], \text{ avec } \phi(\mathbf{x}) = \Pi_{i=1}^{N} \frac{B_{s_i}(x_i)}{s_i!}. \tag{5.20}$$

Propriétés :

1. Si la formule de cubature Q est de degré (polynomial algébrique) d, *i.e.* exacte pour tout polynôme de degré au plus d, alors $B_s = 0$, $s = 1,2,...,d$.
2. Si la formule de cubature Q est symétrique, *i.e.* invariante par symétrie par rapport à $x_i = \frac{1}{2}$ $(i = 1,2,...,N)$, alors $B_s = 0$, $\forall s$ impair. Le développement d'Euler-Maclaurin de $Q^{(m)}[f] - I$ ne contient alors que des puissances paires de $\frac{1}{m}$.
3. Si f est un polynôme de degré $d(f)$, alors $B_s = 0$, $\forall s > d(f)$.

Le corollaire 1 se déduit du théorème 17. Nous considérons une suite d'approximations I_n obtenues en divisant le domaine d'intégration $[0,1]^N$ en hypercubes de côté $\frac{1}{2^n}$ et en utilisant la même méthode de cubature sur chaque sous-domaine.

Corollaire 1 *Soient l'intégrale I et la formule de cubature Q définies par les équations (5.15) et (5.2).*
Soit I_n l'approximation de I obtenue en divisant l'hypercube $[0,1]^N$ en hypercubes de côté $\frac{1}{2^n}$ et en appliquant sur chacun la formule de cubature Q.
Si Q est de degré d, alors

$$I_n - I = \frac{B_{d+1}}{2^{n(d+1)}} + \mathcal{O}\left(\frac{1}{2^{n(d+2)}}\right),$$

où B_{d+1} est défini par l'équation (5.18).

Preuve Puisque $I_n = Q^{(2n)}[f]$, le développement d'Euler-Maclaurin de $I_n - I$ se déduit du théorème 17 en posant $m = 2^n$.
De plus, comme Q est de degré d, d'après la première des trois propriétés énoncées précédemment, les d premiers termes de ce développement s'annulent. □

Le théorème 18 permet, à partir de deux approximations successives I_n et I_{n+1}, de déterminer les premiers chiffres significatifs de la valeur mathématique de l'intégrale. Nous utilisons les mêmes notations et hypothèses que pour le corollaire 1.

Théorème 18

$$C_{I_n,I_{n+1}} = C_{I_n,I} + \log_{10}\left(\frac{2^{d+1}}{2^{d+1}-1}\right) + \mathcal{O}\left(\frac{1}{2^n}\right).$$

La preuve du théorème 18 est similaire à celle du théorème 7 relatif aux méthodes d'approximation. En posant dans l'équation (4.31), à partir de laquelle a été établi le théorème 7, $K = B_{d+1}$, $p = d + 1$ et $q = d + 2$, on retrouve le corollaire 1.

Lorsque la zone de convergence est atteinte, *i.e.* lorsque le terme $\mathcal{O}\left(\frac{1}{2^n}\right)$ devient négligeable, les chiffres significatifs communs à deux approximations successives sont aussi en commun avec la valeur mathématique de l'intégrale, à un bit près. En effet, quel que soit le degré d de la méthode de cubature, le terme $\log_{10}\left(\frac{2^{d+1}}{2^{d+1}-1}\right)$ correspond à un bit.
Si les approximations I_n sont calculées jusqu'à ce que, dans la zone de convergence, la différence $I_n - I_{n+1}$ soit non significative, nous pouvons garantir que les bits significatifs exacts du dernier itéré sont ceux de I, à un près.

Remarque 5 *Si la formule de cubature Q est symétrique, le développement d'Euler-Maclaurin de $Q^{(m)}[f] - I$ donné dans le théorème 17 ne contient que des puissances paires de $\frac{1}{m}$. Le terme d'erreur du corollaire 1 devient alors $\mathcal{O}\left(\frac{1}{2^{n(d+3)}}\right)$ et celui du théorème 18 devient $\mathcal{O}\left(\frac{1}{4^n}\right)$.*

La stratégie décrite pour le calcul d'intégrales sur l'hypercube $[0,1]^N$ peut être généralisée au calcul d'intégrales de la forme

$$J = \int_D g(\mathbf{y})d\mathbf{y}, \tag{5.21}$$

où $D = [a_1,b_1] \times [a_2,b_2] \times ... \times [a_N,b_N]$.

En effet, par changement de variable, nous pouvons ramener J à une intégrale sur l'hypercube $[0,1]^N$:

$$J = \Pi_{i=1}^N (b_i - a_i)I = \Pi_{i=1}^N (b_i - a_i) \int_{[0,1]^N} f(\mathbf{x})d\mathbf{x}, \tag{5.22}$$

où $f(\mathbf{x}) = g\left((b_1 - a_1)x_1 + a_1, (b_2 - a_2)x_2 + a_2,..., (b_N - a_N)x_N + a_N\right)$.

En adoptant les mêmes notations que dans le corollaire 1, nous pouvons approcher l'intégrale J par :

$$J_n = \Pi_{i=1}^N (b_i - a_i)I_n. \tag{5.23}$$

L'erreur de troncature sur J_n est alors :

$$J_n - J = \Pi_{i=1}^N (b_i - a_i)(I_n - I) \tag{5.24}$$

où $I_n - I$ est donné dans le corollaire 1. Comme le facteur $\Pi_{i=1}^N (b_i - a_i)$ ne dépend ni de la formule de cubature utilisée, ni du nombre de découpages du domaine d'intégration, l'équation établie dans le théorème 18 est aussi vérifiée par la suite (J_n).

Remarque 6 *Les résultats établis en 4.5 pour les méthodes d'approximation peuvent aussi s'appliquer au contrôle dynamique des méthodes de cubature. Considérons la suite d'approximations $(Q^{(m)}[f])$ définie par l'équation (5.17).*

- *Si le nombre d'itérations effectuées reste inférieur à une borne qui dépend du degré d de la formule de cubature, alors les bits significatifs communs à deux approximations successives sont aussi communs à la valeur mathématique de l'intégrale, à un près.*

– *Au lieu de comparer deux approximations successives, on peut comparer $Q^{(m)}[f]$ et $Q^{(m+r)}[f]$ ($r \in I\!N^*$) pour obtenir les premiers chiffres significatifs de la valeur exacte de l'intégrale I. Sous les conditions spécifiées en 4.5.2, on peut alors obtenir une approximation dont les bits significatifs exacts sont en commun avec I, à un près.*

5.4 L'atelier ACIM

L'atelier ACIM (Atelier de Calcul d'Intégrales Multiples) a été développé dans le cadre de la thèse de M. Charikhi. Il permet le calcul en ASD d'intégrales de dimension 1 à 3 avec les stratégies décrites dans les chapitres 3 et 4.
L'atelier ACIM comprend :
- une bibliothèque de fonctions
- un logiciel permettant l'utilisation de la bibliothèque via une interface conviviale.

5.4.1 La bibliothèque ACIM

La bibliothèque ACIM est un ensemble de fonctions écrites en Fortran 90, qui peuvent être appelées dans les codes en Fortran 90 ou Fortran 95. ACIM utilise une version gratuite de la bibliothèque CADNA accessible sur internet[2]. La bibliothèque ACIM permet le calcul en double précision d'intégrales de dimension 1 à 3 à bornes fixes ou variables, finies ou infinies.

Trois méthodes de quadrature y ont été implantées : Simpson, Romberg et Gauss-Legendre à 12 points. Elles peuvent être utilisées pour le calcul d'intégrales multiples par le principe des "intégrales itérées".

Les intégrales bidimensionnelles à bornes fixes et finies peuvent aussi être évaluées en utilisant six méthodes de cubature, notées C_Ω^1 à C_Ω^6. Ces méthodes consistent à évaluer une intégrale sur le carré $\Omega = [-1,1]^2$ en utilisant ν points de ce carré. Plus précisément, une intégrale

$$I_\Omega[f] = \int_\Omega f(x,y)\,dx\,dy$$

est approchée par

$$C_\Omega[f] = \sum_{i=1}^{\nu} a_i f(x_i,y_i)$$

où pour $i = 1,...,\nu$, $a_i \in I\!R$ et $(x_i,y_i) \in \Omega$. Un changement de variables est effectué pour l'approximation d'intégrales sur d'autres domaines bidimensionnels.
Les caractéristiques des méthodes C_Ω^1 à C_Ω^3, accessibles dans [1], sont indiquées dans le tableau 5.1. La figure 5.3 présente, pour les méthodes C_Ω^1 à C_Ω^3, la position des ν points (x_i,y_i) sur le carré Ω. Les méthodes C_Ω^4, C_Ω^5 et C_Ω^6, respectivement de degrés 13, 15 et 19, ont été obtenues en appliquant la méthode de Gauss-Legendre, à respectivement 7, 8 et 10 points, à chaque côté du domaine d'intégration.

2. `http://www.lip6.fr/cadna`

méthode	degré	ν	(x_i,y_i)	a_i
C^1_Ω	3	9	(0,0)	16/9
			$(0,\pm 1)$	4/9
			$(\pm 1,0)$	4/9
			$(\pm 1,\pm 1)$	1/9
C^2_Ω	3	4	$(\pm\sqrt{\frac{1}{3}},\pm\sqrt{\frac{1}{3}})$	1
C^3_Ω	5	9	(0,0)	64/81
			$(\pm\sqrt{\frac{3}{5}},\pm\sqrt{\frac{3}{5}})$	25/81
			$(\pm\sqrt{\frac{3}{5}},0)$	40/81
			$(0,\pm\sqrt{\frac{3}{5}})$	40/81

TAB. 5.1 – *Caractéristiques des méthodes C^1_Ω à C^3_Ω*

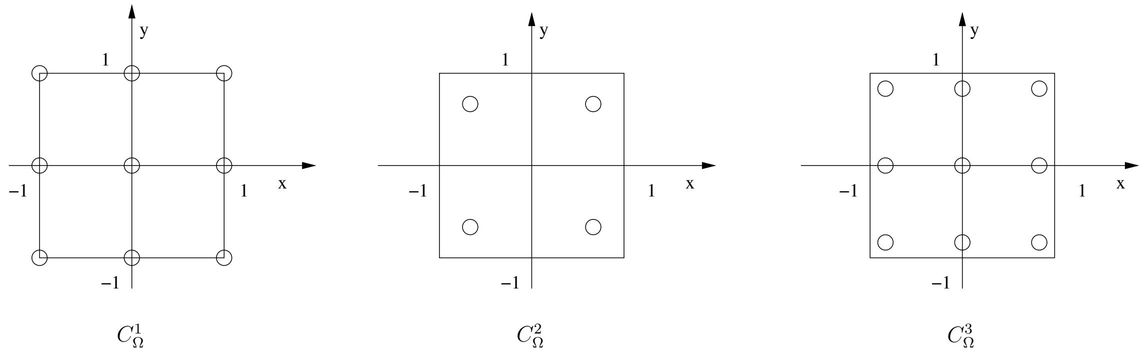

FIG. 5.3 – *Position des ν points (x_i,y_i) sur le carré Ω*

Trois méthodes de cubature ont été implantées dans ACIM pour le calcul d'intégrales tridimensionnelles à bornes fixes et finies. Ces méthodes, notées C^1_Γ à C^5_Γ, consistent à évaluer une intégrale sur le cube $\Gamma = [-1,1]^3$ en utilisant ν points de ce cube. Plus précisément, une intégrale

$$I_\Gamma[f] = \int_\Gamma f(x,y,z)\,dx\,dy\,dz$$

est approchée par

$$C_\Gamma[f] = \sum_{i=1}^{\nu} a_i f(x_i,y_i,z_i)$$

où pour $i = 1,...,\nu$, $a_i \in I\!R$ et $(x_i,y_i,z_i) \in \Gamma$. Un changement de variables est effectué pour l'approximation d'intégrales sur d'autres domaines tridimensionnels.
Les caractéristiques des méthodes C^1_Γ à C^3_Γ, accessibles dans [1], sont indiquées dans le tableau 5.2. La figure 5.4 présente, pour les méthodes C^1_Γ et C^2_Γ, la position des ν points (x_i,y_i,z_i) sur le cube Γ. Les méthodes C^4_Γ et C^5_Γ, respectivement de degrés 9 et 19, ont été obtenues en appliquant la méthode de Gauss-Legendre, à respectivement 5 et 10 points, à chaque côté du domaine d'intégration.

méthode	degré	ν	(x_i,y_i,z_i)	a_i
C_Γ^1	3	6	$(\pm 1,0,0)$	1/6
			$(0,\pm 1,0)$	1/6
			$(0,0,\pm 1)$	1/6
C_Γ^2	5	21	$(0,0,0)$	-496/45
			$(\pm\frac{1}{2},0,0)$	128/45
			$(0,\pm\frac{1}{2},0)$	128/45
			$(0,0,\pm\frac{1}{2})$	128/45
			$(\pm 1,0,0)$	8/45
			$(0,\pm 1,0)$	8/45
			$(0,0,\pm 1)$	8/45
			$(\pm 1,\pm 1,\pm 1)$	5/45
C_Γ^3	7	42	$(\pm 1,0,0)$	728/450
			$(0,\pm 1,0)$	728/450
			$(0,0,\pm 1)$	728/450
			$(0,\pm 1,\pm 1)$	320/450
			$(\pm 1,\pm 1,0)$	320/450
			$(\pm 1,0,\pm 1)$	320/450
			$(\pm 1,\pm\frac{\sqrt{10}}{4},\pm\frac{\sqrt{10}}{4})$	128/450
			$(\pm\frac{\sqrt{10}}{4},\pm 1,\pm\frac{\sqrt{10}}{4})$	128/450
			$(\pm\frac{\sqrt{10}}{4},\pm\frac{\sqrt{10}}{4},\pm 1)$	128/450

TAB. 5.2 – *Caractéristiques des méthodes C_Γ^1 à C_Γ^3*

Quelle que soit la méthode d'intégration choisie, une suite d'approximations est calculée en ASD jusqu'à ce que la différence entre deux itérés successifs soit non significative, conformément aux stratégies décrites auparavant. À partir des chiffres significatifs exacts du résultat (*i.e.* ceux qui ne sont pas entachés d'erreur d'arrondi), on peut alors déterminer ceux qui sont en commun avec la valeur mathématique de l'intégrale.

5.4.2 Le logiciel ACIM

Le logiciel ACIM permet d'utiliser toutes les fonctions de la bibliothèque ACIM via une interface conviviale développée en langage C sous l'environnement Linux. L'utilisateur choisit une des méthodes de quadrature ou de cubature citées en 5.4.1, puis spécifie la fonction à intégrer et les bornes de l'intégrale. La figure 5.5 présente la fenêtre de saisie pour le calcul d'une intégrale double à bornes variables par la méthode de Romberg. Un code en Fortran est généré automatiquement à partir des données fournies par l'utilisateur. Celui-ci appelle la fonction adéquate de la bibliothèque ACIM. Ce code est compilé, puis exécuté. L'approximation obtenue est alors affichée avec ses chiffres significatifs exacts. Un compilateur Fortran 90 ou Fortran 95 est donc nécessaire pour l'utilisation de ce logiciel.

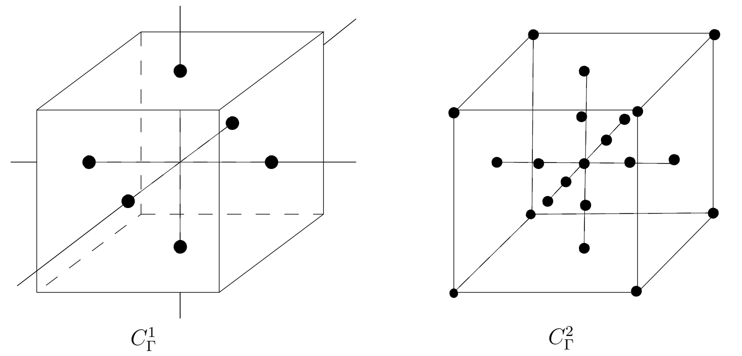

FIG. 5.4 – *Position des ν points (x_i,y_i,z_i) sur le cube Γ*

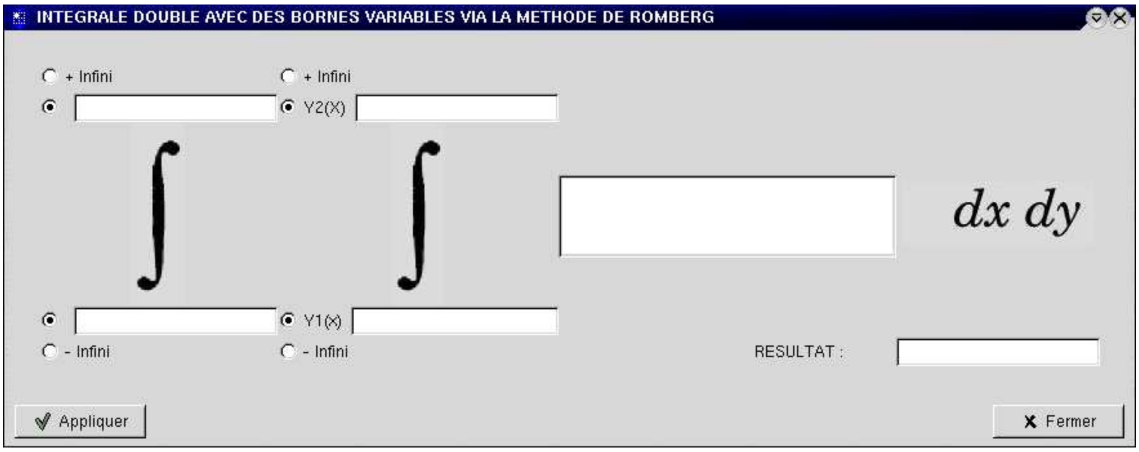

FIG. 5.5 – *Calcul d'intégrale double à bornes variables par la méthode de Romberg*

5.4.3 Expérimentations numériques

Les intégrales suivantes ont été calculées avec la bibliothèque ACIM :

1. $\int_0^1 \int_0^1 y \frac{\arctan(\sqrt{2+x^2})}{(1+x^2)\sqrt{2+x^2}} dxdy = \frac{5\pi^2}{192}$,
2. $\int_0^1 \int_0^1 xy \log(1+y) dxdy = \frac{1}{8}$,
3. $\int_0^1 \int_0^1 \int_0^1 y e^{-\frac{x}{2}} e^{-z} dxdydz = (1-\frac{1}{e})(1-\frac{1}{\sqrt{e}})$,
4. $\int_0^1 \int_0^1 \int_0^1 z \frac{e^{-y}}{1+x} dxdydz = (1-\frac{1}{e})\frac{\log(2)}{2}$.

Le tableau 5.3 présente les approximations obtenues ainsi que les 16 premiers chiffres de la valeur exacte de chaque intégrale. Les temps de calcul ont été mesurés sur un processeur Pentium III à 500 Mhz et sont exprimés en secondes. Les intégrales ont été calculées, d'une part en utilisant le principe des "intégrales itérées" avec la méthode de Simpson, de Romberg ou de Gauss-Legendre à 12 points, d'autre part en utilisant les méthodes de cubature décrites en 5.4.1. Pour chaque résultat obtenu, seuls les chiffres significatifs qui ne sont pas entachés d'erreur d'arrondi sont reportés dans le tableau 5.3.

On remarque que, conformément aux résultats théoriques énoncés auparavant, les chiffres décimaux significatifs exacts des approximations obtenues sont en commun avec la valeur

mathématique de l'intégrale, à un près.
Les méthodes de Romberg et de Gauss-Legendre à 12 points appliquées, grâce au principe des "intégrales itérées", à chaque côté du domaine d'intégration sont à la fois précises et performantes.
Concernant les intégrales bidimensionnelles, la méthode de cubature C_Ω^6, de degré 19, permet d'obtenir des résultats comparables au regard du temps de calcul et de la qualité numérique. Avec l'intégrale 1, on remarque que les méthodes C_Ω^5 et C_Ω^6 fournissent toutes deux un résultat de précision maximale en double précision (15 chiffres significatifs exacts). Le temps d'exécution est alors légèrement plus élevé avec la méthode C_Ω^6, qui nécessite plus d'évaluations de la fonction à intégrer.
Concernant les intégrales tridimensionnelles, outre les méthodes de Romberg et de Gauss-Legendre à 12 points appliquées trois fois, les méthodes de cubature C_Γ^4 et C_Γ^5, respectivement de degrés 9 et 19, permettent un calcul à la fois fiable et performant des intégrales étudiées. Avec l'intégrale 3, l'approximation obtenue par la méthode C_Γ^4 (à 125 points) est de précision maximale (15 chiffres significatifs exacts), celle obtenue par la méthode C_Γ^5 (à 1000 points) a un chiffre significatif exact de moins. En revanche, avec l'intégrale 4, la méthode C_Γ^5 est plus appropriée que la méthode C_Γ^4, puisqu'elle fournit une approximation de précision maximale et requiert un temps de calcul relativement faible.
Le principe des "intégrales itérées" peut être particulièrement avantageux en ASD, puisqu'il permet de déterminer le pas d'intégration optimal pour chaque dimension du domaine considéré.

5.4.4 Comparaison avec Cubpack++

La bibliothèque Cubpack++ [53], développée en C++, permet l'approximation d'intégrales sur une grande variété de régions bidimensionnelles. Cette bibliothèque utilise une méthode de cubature de degré 13 à 37 points pour l'évaluation d'intégrales sur un domaine rectangulaire. On peut alors spécifier le nombre maximal d'évaluations (fixé à 10^8 dans nos expérimentations numériques) et l'erreur relative souhaitée. Le résultat est toujours fourni avec une estimation de l'erreur de troncature commise.

Considérons les intégrales suivantes.

1. $\int_0^{10}\int_0^{10} \arctan(\sin(1.3x)\cos(3.2y))dxdy$
2. $\int_0^{1.2}\int_0^{1.2} \exp(-2.99x)\,|\cos(1.3y)|dxdy$
3. $\int_{-5}^{1}\int_{-5}^{5} \cos(2.99x)\sin(xy)f(x)f(y)dxdy$, avec $f(t) = 20\cos(20t)((2.7t-3.3)t+1.2)$
4. $\int_{-4}^{6.2}\int_{-4}^{5.2} \frac{5.14+\sin(\cosh(xy))}{\cos(xy)^2+199.29}dxdy$

Le tableau 5.4 permet de comparer les résultats de Cubpack++ avec ceux obtenus par la méthode C_Ω^4 d'ACIM, elle aussi de degré 13. En outre une approximation de l'intégrale 2 a pu être obtenue avec le logiciel Maple [136].
Seuls les chiffres significatifs exacts (*i.e.* non entachés d'erreur d'arrondi) des approximations fournies par ACIM sont reportés dans le tableau 5.4. D'après les résultats énoncés en 5.3, les chiffres significatifs exacts des approximations obtenues avec la méthode C_Ω^4 d'ACIM sont ceux de la valeur mathématique de l'intégrale, à un près.

On remarque que *Cubpack++* peut dépasser légèrement le nombre maximum d'évaluations imposé par les paramètres d'entrée de la fonction *Integrate* de cette bibliothèque. Dans ce cas, l'erreur estimée peut être supérieure à l'erreur passée en paramètre.
Si l'on compare les chiffres fournis par Cubpack++ avec ceux obtenus avec ACIM, la différence se situe au pire à l'avant-dernier chiffre.
Avec toutes les intégrales calculées pour cette expérimentation, si l'on tient compte de l'erreur estimée par Cubpack++, pour obtenir la même précision qu'avec ACIM, Cubpack++ nécessite bien plus d'évaluations de la fonction à intégrer. La bibliothèque ACIM est alors plus performante, même si une fonction de type stochastique est environ trois fois plus coûteuse que la fonction correspondante de type flottant.

5.5 Calcul d'une intégrale issue de la théorie des étoiles à neutrons

L'étude décrite ici est décrite en détail en 8.9. Il s'agit de contrôler dynamiquement le calcul d'une intégrale multiple impropre issue de la théorie des étoiles à neutrons. Nous considérons l'intégrale suivante, qui a été établie par l'astro-physicien L. Villain [184].

$$\tau\left(\varepsilon,v\right) = \frac{1}{\omega\left(\varepsilon\right)} \int_0^{\frac{\pi}{2}} d\theta \sin\left(\theta\right) \int_0^{\infty} dn\, n^2 \int_0^{\infty} dp\, h\left(n,p,\theta,\varepsilon,v\right), \text{ où } (\varepsilon,v) \in [10^{-4},10^4]\times[10^{-4},10^3]$$

avec $h\left(n,p,\theta,\varepsilon,v\right) = \psi(z)\Gamma(n-\varepsilon-z)+\psi(-z)\Gamma(n-\varepsilon+z)-\psi(z)\Gamma(n+\varepsilon-z)-\psi(z)\Gamma(n+\varepsilon+z)$, $z = \sqrt{p^2 + (v\sin(\theta))^2}$, $\psi(x) = \frac{1}{\exp x+1}$, $\Gamma(x) = \frac{x}{\exp x-1}$ et $\omega(\varepsilon) = \frac{17\pi^4}{60}\varepsilon\left(1 + \frac{10\varepsilon^2}{17\pi^2} + \frac{\varepsilon^4}{17\pi^4}\right)$.

ω est une fonction de normalisation, définie telle que $\tau\left(\varepsilon,0\right) = 1$.

Le but est de calculer l'intégrale $\tau\left(\varepsilon,v\right)$ pour un tableau de valeurs (ε,v) et d'effectuer une interpolation pour les autres, afin de déterminer la surface $\tau\left(\varepsilon,v\right)$.

Plusieurs problèmes numériques peuvent survenir lors du calcul de $\tau\left(\varepsilon,v\right)$.

– L'intégrale multiple $\tau\left(\varepsilon,v\right)$ peut se décomposer en plusieurs intégrales monodimensionnelles, afin d'être évaluée grâce au principe des "intégrales itérées".
  En effet,
  $$\tau\left(\varepsilon,v\right) = \frac{1}{\omega\left(\varepsilon\right)} \int_0^{\frac{\pi}{2}} \sin\left(\theta\right) d\theta \int_0^{\infty} n^2 dn \int_0^{\infty} h\left(\varepsilon,v,\theta,n,p\right) dp.$$
  Par conséquent,
  $$\tau\left(\varepsilon,v\right) = \frac{1}{\omega\left(\varepsilon\right)} \int_0^{\frac{\pi}{2}} \sin(\theta) f(\varepsilon,v,\theta)\, d\theta$$
  avec
  $$f(\varepsilon,v,\theta) = \int_0^{\infty} n^2 g(\varepsilon,v,\theta,n) dn$$
  et
  $$g(\varepsilon,v,\theta,n) = \int_0^{\infty} h\left(\varepsilon,v,\theta,n,p\right) dp.$$

Une méthode de quadrature peut donc être utilisée. La méthode de Gauss-Legendre a été choisie pour son erreur de troncature relativement faible. Néanmoins, comme toute méthode d'approximation, elle génère à la fois une erreur de troncature et une erreur d'arrondi qui doivent être contrôlées.

- Les intégrales $f(\varepsilon,v,\theta)$ et $g(\varepsilon,v,\theta,n)$ ont toutes deux une borne infinie, qui doit être remplacée par une borne finie suffisamment élevée. Le choix d'une valeur optimale pour ces deux bornes peut alors s'avérer difficile.
- Le calcul de $\Gamma(x) = x/(\exp(x) - 1)$ peut générer des cancellations lorsque x est proche de 0. Néanmoins on peut utiliser un développement en série de Γ.
  En effet, $\Gamma(x) \approx 1/\left(1 + \frac{x}{2} + ... + \frac{x^{k-1}}{k!}\right)$.
  Cependant le choix d'une valeur optimale pour le paramètre k reste problématique.

Nous présentons en 8.9 comment contrôler dynamiquement le calcul de l'intégrale $\tau\,(\varepsilon,v)$, afin d'obtenir une approximation qui minimise l'erreur globale. De plus, en composant différents résultats théoriques décrits précédemment, nous montrons que les bits significatifs exacts de l'approximation obtenue sont ceux de $\tau\,(\varepsilon,v)$, à six près. Comme six bits correspondent à 1.8 chiffres décimaux, on peut alors garantir que les chiffres décimaux significatifs exacts de l'approximation obtenue sont en commun avec la valeur exacte, à deux près.

En utilisant la stratégie décrite en 8.9, l'intégrale $\tau\,(\varepsilon,v)$ a été calculée pour 5752 points (ε,v) définis par :

$$\left\{\begin{array}{lllll} \varepsilon & = & 10^a & \text{avec} & a = -4.0, -3.9, -3.8, \ldots, 4.0 \\ v & = & 10^b & \text{avec} & b = -4.0, -3.9, -3.8, \ldots, 3.0. \end{array}\right.$$

Les intégrales impropres $f(\varepsilon,v,\theta) = \int_0^\infty n^2 g(\varepsilon,v,\theta,n)dn$ et $g(\varepsilon,v,\theta,n) = \int_0^\infty h\,(\varepsilon,v,\theta,n,p)\,dp$ ont été approchées par des intégrales sur un intervalle fini dont la borne supérieure L_i a pu être déterminée dynamiquement. La suite (L_i) qui a été choisie est définie par :

$$\left\{\begin{array}{lll} L_i & = & 10i \qquad (i = 0,...,20) \\ L_i & = & 50i - 800 \qquad (i = 21,...,36) \\ L_i & = & 250i - 8000 \qquad (i = 37,...,52) \\ L_i & = & 1250i - 60000 \qquad (i \geq 53). \end{array}\right.$$

Chaque intégrale a été approchée en utilisant la méthode de Gauss-Legendre à 12 points sur des sous-intervalles du domaine d'intégration. Le nombre q_i de sous-intervalles a pu être déterminé dynamiquement. La suite (q_i) qui a été choisie est définie par :

$$\left\{\begin{array}{lll} q_i & = & i \qquad (i = 1,...,8) \\ q_i & = & 2i - 8 \qquad (i = 9,...,12) \\ q_i & = & 2q_{i-1} \qquad (i \geq 13). \end{array}\right.$$

D'après 4.5.2, si la zone de convergence est atteinte, les bits significatifs communs à l'approximation sur q_i sous-intervalles et à l'approximation sur q_{i+1} sous-intervalles sont aussi communs à la valeur exacte, à $\delta_i = -\log_2\left(1 - \left(\frac{q_i}{q_{i+1}}\right)^{24}\right)$ près. Comme $\forall i,\ \delta_i \leq 1$, alors, dans la zone de convergence, les bits significatifs communs à deux approximations successives sont aussi communs à la valeur exacte, à un près.

L'intégrale $\tau(\varepsilon,v)$ a été calculée en simple précision en utilisant l'ASD implantée dans la bibliothèque CADNA. Le code source peut être consulté sur internet [3]. Le temps d'exécution du code varie de 45 s à 3347 s selon les valeurs de ε et v, le temps d'exécution moyen étant 389 s. Le temps d'exécution est bien plus important en double précision. En effet, toutes les suites sont calculées jusqu'à ce que la différence entre deux itérés successifs soit due aux erreurs d'arrondi. Le nombre d'itérations effectuées pour que ce test d'arrêt soit satisfait est donc bien plus élevé en double précision.

Les calculs sont relativement stables. En effet, le nombre de chiffres significatifs exacts des approximations de $\tau(\varepsilon,v)$ obtenues, *i.e.* le nombre de chiffres significatifs non entachés d'erreur d'arrondi, varie de 3 à 7. Le tableau 5.5 présente l'occurrence de chaque précision parmi les 5752 résultats obtenus. On remarque que la majorité des approximations ont 6 ou 7 chiffres significatifs exacts. Cette précision est très satisfaisante pour des calculs effectués en simple précision. La précision minimale (3 chiffres significatifs exacts) a été obtenue seulement une fois, pour l'approximation de $\tau(10^{-4},10^{2.9}) \approx 10^{-5}$. D'après les résultats établis en 8.9, les chiffres significatifs exacts de chaque approximation obtenue sont en commun avec la valeur exacte correspondante, à deux près. Par conséquent, d'après le tableau 5.5, on peut garantir de 1 à 5 chiffres significatifs dans les résultats obtenus. La figure 5.6 présente les approximations de $\tau(\varepsilon,v)$ qui ont été obtenues.

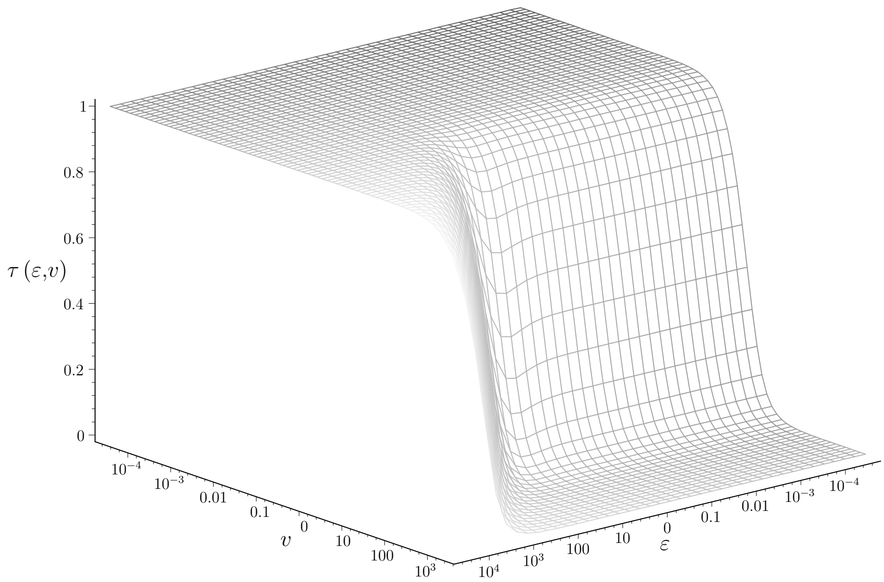

FIG. 5.6 – *Résultats numériques*

L'astro-physicien L. Villain, qui a établi l'intégrale $\tau(\varepsilon,v)$, nous a suggéré le calcul d'autres intégrales multiples issues de la théorie des étoiles à neutrons. L'intégrale $\tau(\varepsilon,v)$ a été générée à partir d'une intégrale $\tau(\varepsilon,v,w)$, en supposant le paramètre w négligeable par rapport à v. La prise en compte de ce troisième paramètre constitue donc une suite possible à

3. http://www-anp.lip6.fr/cadna/Examples_Dir/Star/Accueil.php

notre collaboration. Une perspective plus problématique est le calcul d'une intégrale de dimension 11 issue d'une généralisation de l'intégrale $\tau(\varepsilon,v,w)$.

5.6 Conclusion et perspectives

En effectuant un contrôle dynamique du calcul d'une intégrale multiple, on peut, comme avec les méthodes numériques citées au chapitre précédent, obtenir une approximation pour laquelle l'erreur globale est minimale. Dans le cas d'une intégrale non singulière sur un domaine fini, les chiffres significatifs exacts (non entachés d'erreur d'arrondi) de l'approximation obtenue sont en général ceux de la valeur mathématique, à un près.

Les stratégies décrites en 5.2 permettent, grâce au principe des "intégrales itérées", de contrôler dynamiquement le pas d'intégration pour chaque dimension du domaine traité. On peut alors déterminer le nombre de bits significatifs exacts du résultat qui sont en commun avec la valeur mathématique de l'intégrale. Ce nombre dépend de la dimension du domaine d'intégration, du nombre de bornes impropres, de l'éventuelle approximation de la fonction à intégrer par son développement en série, de la vitesse de convergence des différentes suites mises en œuvre. Il est donc parfois nécessaire d'effectuer une étude spécifique à l'intégrale, telle celle décrite en 5.5.

Lors du calcul d'une intégrale non singulière sur un hyperrectangle, la stratégie décrite en 5.3 est fondée sur l'application d'une méthode de cubature à des sous-domaines de même taille. Le nombre optimal de découpages effectués peut alors être contrôlé dynamiquement. Grâce au développement d'Euler-Maclaurin de l'erreur de troncature commise, nous avons montré que les bits significatifs exacts de l'approximation obtenue sont ceux de la valeur mathématique de l'intégrale, à un près. Ce résultat pourrait être étendu à d'autres types d'intégrales en utilisant le développement de l'erreur de troncature générée. Le développement d'Euler-Maclaurin a d'ailleurs été établi pour les simplexes de dimension N [134]. En outre, plusieurs études ont porté sur le développement de l'erreur commise lors du calcul d'intégrales singulières [130, 132, 165, 174].

Les stratégies utilisées dans la bibliothèque Cubpack [52] sont elles aussi fondées sur des découpages du domaine d'intégration. Le nombre de découpages effectués dépend alors de l'estimation de l'erreur de troncature commise : celle-ci doit respecter un seuil fixé par l'utilisateur. Dans la dernière version de la bibliothèque Cubpack, il est possible d'accélérer la convergence de la suite d'approximations générées grâce à l'ε-algorithme de Wynn [50]. Les travaux de R. Adout sur l'application d'algorithmes d'extrapolation aux méthodes de quadrature [3] pourraient être étendus aux méthodes de cubature. Il serait intéressant d'étudier l'impact des algorithmes d'extrapolation sur la convergence des méthodes de cubature, mais aussi sur la qualité numérique des résultats obtenus.

Pour le calcul d'une intégrale sur un hyperrectangle de dimension N, la stratégie décrite en 5.3 consiste à effectuer à chaque itération 2^N découpages de chaque sous-domaine. À l'itération n, la méthode de cubature doit donc être appliquée à $(2^N)^n$ sous-domaines. Les stratégies adaptatives, en tenant compte du comportement numérique de la fonction à intégrer, permettent d'éviter le découpage de certains sous-domaines. De nombreuses procédures adaptatives

(HALF [63], ADAPT [74], DCUHRE [21, 22]) effectuent, non pas 2^N, mais 2 découpages des sous-domaines. Un sous-domaine est alors partitionné en 2 moitiés dans la direction où la différence divisée d'ordre 4 est la plus élevée. Une autre stratégie adaptative décrite dans [50, 54] a été implantée dans la dernière version de Cubpack. Celle-ci permet de déterminer, non seulement la direction des découpages, mais aussi, pour chaque sous-domaine, le nombre de découpages, de la forme 2^i $(1 \leq i \leq N)$. Il serait intéressant d'établir une stratégie adaptative qui tienne compte de la propagation d'erreur d'arrondi et d'effectuer un contrôle dynamique du nombre de découpages de chaque sous-domaine.

Les méthodes de Monte Carlo ou de quasi-Monte Carlo [28, 143, 148] sont utilisées pour l'évaluation d'intégrales de dimension élevée. Cependant le résultat obtenu est en général de précision médiocre. Bien que de nombreux travaux portent sur l'estimation de l'erreur de méthode commise, les apports de l'ASD dans ce domaine constituent un axe de recherche à explorer.

intégrale	méthode	approximation	tps de calcul (s)
	valeur exacte	0.257020947945035E +000	-
1	Simpson	0.25702094794503E +000	8.0E+00
	Romberg	0.257020947945035E +000	2.1E-02
	Gauss-Legendre	0.257020947945035E +000	1.9E-02
	C_Ω^1	0.2570209479450E +000	1.0E+01
	C_Ω^2	0.25702094794503E +000	1.7E+01
	C_Ω^3	0.25702094794503E +000	2.3E+00
	C_Ω^4	0.257020947945035E +000	5.E-02
	C_Ω^5	0.257020947945035E +000	2.0E-02
	C_Ω^6	0.257020947945035E +000	3.0E-02
	valeur exacte	0.125000000000000E +000	-
2	Simpson	0.124999999999999E+000	8.9E-02
	Romberg	0.124999999999999E+000	1.0E-03
	Gauss-Legendre	0.124999999999999E+000	1.9E-03
	C_Ω^1	0.1250000000000E+000	1.6E+01
	C_Ω^2	0.1249999999999E+000	3.4E+01
	C_Ω^3	0.1250000000000E+000	1.0E+00
	C_Ω^4	0.125000000000000E +000	2.5E-02
	C_Ω^5	0.124999999999999E +000	8.2E-03
	C_Ω^6	0.125000000000000E +000	3.0E-03
	valeur exacte	0.248720059264354E+000	-
3	Simpson	0.24872005926435E+000	6.4E+01
	Romberg	0.248720059264354E+000	6.9E-02
	Gauss-Legendre	0.248720059264354E+000	3.9E-01
	C_Γ^1	0.2487200592642E+000	1.1E+03
	C_Γ^2	0.2487200592643E+000	4.1E+01
	C_Γ^3	0.2487200592643E+000	8.2E+01
	C_Γ^4	0.248720059264353E+000	7.0E-02
	C_Γ^5	0.24872005926435E+000	7.0E-02
	valeur exacte	0.219076291562995E+000	-
4	Simpson	0.21907629156299E+000	3.2E+02
	Romberg	0.219076291562995E+000	7.0E-01
	Gauss-Legendre	0.219076291562995E+000	3.0E-01
	C_Γ^1	0.21907629156E+000	3.5E+05
	C_Γ^2	0.2190762915629E+000	3.0E+01
	C_Γ^3	0.219076291562E+000	4.7E+02
	C_Γ^4	0.21907629156299E+000	4.0E+00
	C_Γ^5	0.219076291562996E+000	5.0E-02

TAB. 5.3 – *Calcul d'intégrales par la bibliothèque ACIM*

int.	méthode	err relative souhaitée	approximation	nb d'éval.	err relative estimée
1	ACIM : C_Ω^4	-	0.1195832831077E-001	267589	-
	Cubpack++	1.E-10	0.119583283107737E-001	408739	8.E-11
		1.E-12	0.119583283107731E-001	100000011	3.E-11
2	Maple	-	0.2501388601895579E+000	-	-
	ACIM : C_Ω^4	-	0.250138860189557E+000	1029	-
	Cubpack++	1.E-10	0.250138860189558E+000	1739	3.E-13
		1.E-12	0.250138860189558E+000	1739	3.E-13
		1.E-14	0.250138860189571E+000	100000011	1.E-14
3	ACIM : C_Ω^4	-	0.531892148039E+003	1070405	-
	Cubpack++	1.E-10	0.53189214803942E+003	8665881	1.E-11
		1.E-12	0.53189214803942E+003	100000011	4.E-12
4	ACIM : C_Ω^4	-	0.25072E+001	66885	-
	Cubpack++	1.E-4	0.250752917598152E+001	100000011	6.E-02

TAB. 5.4 – *Calcul d'intégrales par les bibliothèques ACIM et Cubpack++*

nb. de chiffres significatifs exacts	occurrence
3	1
4	217
5	665
6	3347
7	1522

TAB. 5.5 – *Qualité numérique de $\tau\,(\varepsilon,v)$*

Chapitre 6

Conclusion et perspectives

Après un rappel des principaux résultats établis dans ce document, nous présentons quelques axes de recherche futurs qui peuvent revêtir des aspects aussi bien théoriques que pratiques.

Au cours de ces dernières années, l'Arithmétique Stochastique Discrète (ASD) a bénéficié de différents développements avec deux principaux objectifs : faciliter son utilisation, améliorer ses performances. Le premier objectif a suscité la réalisation de divers outils : celui qui modifie automatiquement le code d'un utilisateur pour être exécuté en ASD, celui qui analyse automatiquement les instabilités numériques générées durant l'exécution, celui qui permet de visualiser la qualité numérique de résultats volumineux. La validation numérique ne pouvant être acceptée qu'à un coût modéré, des efforts ont été accomplis dans ce sens pour l'implémentation de l'ASD sur différentes architectures : séquentielles, vectorielles et parallèles. L'ASD a été implantée sur deux machines vectorielles : NEC SX5 et CRAY SV1. Le premier calculateur respecte la norme IEEE 754, mais pas le second. Mettre en œuvre le mode d'arrondi aléatoire sans inhiber la vectorisation a donc nécessité des développements spécifiques à chaque machine. Des surcharges d'opérateurs ont été écrites afin que les instructions utilisant la notation “tableau” du Fortran 90 soient vectorisées. Il est apparu que les performances de l'ASD dépendent fortement du compilateur, notamment de ses possibilités d'*inlining* et de sa gestion de la mémoire.

Les algorithmes décrits dans les chapitres 4 et 5 sont tous fondés sur un calcul de suite. Ce calcul peut être intrinsèque à la méthode utilisée. Par exemple, la méthode de Romberg consiste à calculer une suite d'approximations d'une intégrale. Lorsqu'une méthode numérique est fondée sur un pas de discrétisation, une suite d'approximations peut être générée grâce à une diminution régulière de ce pas. Une telle suite peut aussi être créée en effectuant des subdivisions successives du domaine traité auxquelles on applique la même méthode numérique.

Les résultats théoriques établis dans les chapitres 4 et 5 permettent, à partir de deux itérés d'une suite dans la zone de convergence, de déterminer les premiers chiffres de la limite exacte. Ces résultats théoriques deviennent particulièrement intéressants si l'on peut estimer la propagation d'erreur d'arrondi, par exemple grâce à l'ASD. Si les calculs sont effectués jusqu'à ce que, dans la zone de convergence, la différence entre deux itérés successifs soit non significative, alors l'erreur globale sur le dernier itéré est minimale. On peut alors déterminer les chiffres significatifs du résultat obtenu qui sont en commun avec la limite exacte.

Les différents théorèmes énoncés dans ce document peuvent se combiner et ainsi s'appliquer aux algorithmes numériques nécessitant le calcul de plusieurs suites. Le théorème 1, établi au chapitre 2, est alors primordial puisqu'il permet, lorsqu'une opération arithmétique est effectuée en ASD entre plusieurs suites, de déterminer les chiffres significatifs du résultat en commun avec la valeur exacte correspondante. Toutefois les résultats théoriques établis dans ce document ne dispensent pas d'une étude préalable du problème traité. En effet, les informations relatives à la convergence des suites mises en œuvre sont fondamentales : le type de convergence (monotone ou alternée) et la vitesse de convergence (linéaire ou exponentielle).

Deux applications sont décrites dans ce document. La première consistait à calculer par méthode de quadrature une intégrale impropre établie en cristallographie. La stratégie proposée nous a permis d'effectuer un contrôle dynamique du calcul de l'intégrale par la méthode de Romberg. Les approximations calculées en ASD ont alors pu être comparées avec celles obtenues en effectuant un développement en série de l'intégrale. La seconde application, plus complexe, consistait à calculer une intégrale multiple impropre issue de l'étude des étoiles à neutrons, pour laquelle aucune expression analytique n'a pu être établie. Pour chaque suite mise en œuvre, le test d'arrêt utilisé a permis d'obtenir l'itéré pour lequel l'erreur globale est minimale. Le nombre d'itérations effectuées et, par conséquent, le temps de calcul dépendent de la précision choisie. Compte tenu de la stabilité du calcul, il a alors été préférable d'exécuter le code en simple précision.

Les stratégies décrites dans ce document ont été mises en œuvre en simple ou en double précision. Il serait intéressant de pouvoir spécifier le nombre de chiffres du résultat que l'on souhaite communs avec la valeur exacte. Ceci est déjà possible lors du calcul d'une suite : en comparant deux itérés, on peut déterminer les k premiers chiffres de la limite exacte. En s'assurant que l'erreur d'arrondi reste inférieure à l'erreur de méthode, ces k chiffres peuvent être obtenus à l'issue d'un calcul effectué en arithmétique à virgule flottante. Cependant le théorème 1, utilisé lorsque plusieurs suites sont combinées entre elles, s'applique aux calculs effectués en ASD. Ce théorème nécessite l'estimation du nombre de chiffres significatifs exacts (*i.e.* non entachés d'erreur d'arrondi) des résultats et ce nombre ne peut être fixé a priori. Un travail théorique doit donc être effectué afin de pouvoir spécifier le nombre de chiffres que l'on souhaite garantir à l'issue d'un calcul combinant plusieurs algorithmes d'approximation.

Les stratégies décrites dans les chapitres 4 et 5 consistent à calculer chaque suite mise en œuvre jusqu'à ce que la différence entre deux itérés successifs soit non significative. Une intégrale sur un domaine infini peut être approchée par une somme d'intégrales sur des domaines finis. Avec la stratégie décrite en 4.4.2, l'erreur globale sur chaque intégrale est minimale. Or, lorsqu'une intégrale est d'un ordre de grandeur très inférieur au total en cours d'évaluation, seuls ses premiers chiffres significatifs exacts sont pris en compte. D'autres tests d'arrêt peuvent être utilisés afin de ne pas calculer certaines intégrales avec une précision inutilement élevée. Il faudrait donc établir de nouveaux théorèmes permettant de déterminer le nombre de chiffres significatifs en commun entre l'approximation obtenue avec ces tests d'arrêt et la valeur exacte correspondante.

Les performances, sans cesse croissantes, des moyens de calcul nécessitent de poursuivre les travaux relatifs à l'implémentation des méthodes automatiques d'analyse d'erreur d'arrondi. Afin de permettre une estimation des erreurs d'arrondi à un coût modéré en temps d'exécution, il faut s'adapter à l'évolution des architectures (séquentielles, vectorielles ou parallèles), des compilateurs, des processeurs. Plusieurs portages de l'ASD peuvent être envisagés. Une bibliothèque pourrait être développée pour les codes s'exécutant avec le logiciel MATLAB. Une bibliothèque implantant l'arithmétique d'intervalles a d'ailleurs déjà été écrite pour MATLAB [80, 163]. Une bibliothèque implantant l'ASD dans les codes en Java pourrait également être développée. Une telle implémentation a déjà été effectuée en arithmétique d'intervalles [20]. Développer un compilateur intégrant les types stochastiques et les fonctionnalités de l'ASD est une autre possibilité. Un compilateur Fortran 95 intègre déjà l'arithmétique d'intervalles [173]. Pour permettre l'utilisation du mode d'arrondi aléatoire, indispensable en ASD, des procédures spécifiques à certains types de processeurs ont été développées en assembleur. Le portage de l'ASD sur le processeur Itanium constitue un axe de recherche qui pourrait éventuellement bénéficier des études menées pour l'implémentation de la méthode CENA sur ce processeur [121].

Les bibliothèques d'algèbre linéaire sont très utilisées dans les codes de calcul scientifique. La bibliothèque LAPACK [13] calcule des bornes d'erreur sur les résultats fournis grâce à l'analyse inverse (décrite en 2.1.2). Une proposition [57] porte sur l'implémentation des BLAS (procédures basiques d'algèbre linéaire) en arithmétique d'intervalles. Il serait intéressant de développer une bibliothèque d'algèbre linéaire qui intègre les fonctionnalités de l'ASD pour l'estimation d'erreur d'arrondi et le contrôle dynamique des calculs.

Les grilles de calcul sont de plus en plus utilisées pour la simulation numérique. Un code peut alors être exécuté sur des machines hétérogènes avec des arithmétiques parfois différentes. L'identité de ces machines n'est pas toujours connue des utilisateurs. Il peut être difficile d'implanter une méthode automatique d'analyse d'erreurs d'arrondi sur une telle grille. Afin de renseigner l'utilisateur sur la qualité numérique de son code, il faudrait alors proposer de nouvelles méthodologies qui pourraient tirer parti des éventuelles différences constatées sur les résultats d'une exécution à l'autre.

L'adaptabilité est nécessaire pour développer un logiciel d'analyse des erreurs d'arrondi dans un environnement technique en perpétuelle évolution. Néanmoins certains choix lors de la conception de processeurs ou de compilateurs peuvent être suggérés par la communauté de l'arithmétique des ordinateurs. Bien que la norme IEEE 754 de 1985 pour l'arithmétique à virgule flottante soit aujourd'hui largement répandue, celle-ci est incomplète. Une proposition a été rédigée afin d'améliorer l'implémentation des fonctions mathématiques élémentaires [61].

L'étude de la qualité numérique de codes de calcul scientifique a donné lieu à plusieurs collaborations, notamment avec l'IPSL (Institut Pierre Simon Laplace) pour l'estimation de la propagation d'erreur d'arrondi dans des codes simulant la circulation océanique [23, 24, 93]. Un projet est actuellement mené pour le contrôle de la qualité numérique de codes de physique atomique développés à l'université de Belfast. Une plus large diffusion des travaux effectués dans le domaine de l'arithmétique des ordinateurs serait bénéfique pour la sécurité des codes de simulation numérique.

Chapitre 7

Bibliographie

[1] M. Abramowitz and I.A. Stegun. *Handbook of mathematical functions*, volume 55 of *Applied Mathematics Series.* Washington : National Bureau of Standards, 1964. reprinted 1968 by Dover Publications, New York.

[2] E. Adams and U. Kulisch, editors. *Scientific computing with automatic result verification*, volume 189 of *Mathematics in Science and Engineering.* Academic Press, San Diego, 1993.

[3] R. Adout. Validation numérique des méthodes d'extrapolation. rapport de stage de DEA Informatique et Recherche Opérationnelle, Université Paris 6, September 2002.

[4] R. Adout, J.-M. Chesneaux, and F. Jézéquel. A reliable hybrid extrapolation algorithm to accelerate the restarted GMRES method. In *Proc. SCAN2004 conference, Fukuoka, Japan*, October 2004.

[5] Z. Ahmed. Definitely an integral. *American Mathematical Monthly*, 109(7):670–671, 2002.

[6] G. Alefeld and J. Herzberger. *Introduction to interval analysis.* Academic Press, 1983.

[7] R. Alt. Floating-point error propagation in iterative methods. *Math. Comput. Simulation*, 30(6):505–517, 1988.

[8] R. Alt. Parallel methods across time for initial value problems. In *Proc. 3rd Internat. Colloquium on Numer. Anal.*, pages 223–230, Plovdiv, Bulgaria, 1994. VSP Internat. Sc. Pub.

[9] R. Alt. On the optimal solution of collocation methods for ODEs. In *Proc. CESA'96 Multiconference on computational engineering in systems and applications*, pages 1128–1133, Lille, France, 1996.

[10] R. Alt and J.-L. Lamotte. Parallel integration across time of initial value problems using PVM. In *Proc. EuroPVM'96 conference*, Munich, Germany, 1996.

[11] R. Alt and J.-L. Lamotte. Experiments on the evaluation of functional ranges using a random interval arithmetic. *Math. Comput. Simulation*, 56:17–34, March 2001.

[12] R. Alt and J. Vignes. Validation of results of collocation methods for ODEs with the CADNA library. *Applied Numerical Mathematics*, 21(2):119–139, 1996.

[13] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users'*

Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 3rd edition, 1999.

[14] A. Asserrhine. *Implémentation de l'arithmétique aléatoire sur machines distribuées et application.* PhD thesis, Université P. et M. Curie, Paris, France, May 1996.

[15] A. Asserrhine, J.-M. Chesneaux, and J.-L. Lamotte. Estimation of round-off errors on several computers architectures. *J. of Universal Computer Science*, 1(7):454–468, July 1995.

[16] R. Avot-Chotin, J.-M. Chesneaux, and J.-L. Lamotte. On the computation of the CESTAC function. In *Proc. 5th Real Numbers and Computers conference*, Lyon, France, September 2003.

[17] D.H. Bailey and X.S. Li. A comparison of three high-precision quadrature schemes. In *Proc. 5th Real Numbers and Computers conference*, pages 81–95, Lyon, France, September 2003.

[18] C.T.H. Baker and G.S. Hodgson. Asymptotic expansions for integration formulas in one or more dimensions. *SIAM J. Numer. Anal.*, 8:473–480, 1971.

[19] F.L. Bauer, H. Rutishauser, and E. Stiefel. New aspects in numerical quadrature. In *Proc. Sympos. Appl. Math.*, volume XV, pages 199–218. Amer. Math. Soc., Providence, R.I., 1963.

[20] B.R. Callajas Bedregal and J. Enéas Montenegro Dutra. An intervalar library in Java. In *SCAN2000, 9th International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics, Karlsruhe, Germany.* GAMM-IMACS, September 2000.

[21] J. Berntsen, T.O. Espelid, and A. Genz. Algorithm 698: DCUHRE - an adaptive multidimensional integration routine for a vector of integrals. *ACM Trans. Math. Softw.*, 17:452–456, 1991.

[22] J. Berntsen, T.O. Espelid, and A. Genz. An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Trans. Math. Softw.*, 17:437–451, 1991.

[23] F. Bonnet-Terras. Etude de la stabilité numérique du code ORCA. Note technique du pôle de modélisation de l'IPSL, Institut Pierre Simon Laplace, October 1999. Available at `http://www.ipsl.jussieu.fr/modelisation/liste-notes-tech.html`.

[24] F. Bonnet-Terras, M.-A. Foujols, and M. Lévy. Etude de la stabilité numérique du code OPA-TRC-NNPZDDOM. Note technique du pôle de modélisation de l'IPSL, Institut Pierre Simon Laplace, June 2000. Available at `http://www.ipsl.jussieu.fr/modelisation/liste-notes-tech.html`.

[25] C. Brezinski and M. Redivo Zaglia. *Extrapolation Methods, theory and Practice.* North Holland, 1991.

[26] R.L. Burden and J.D. Faires. *Numerical analysis.* Brooks-Cole Publishing, 7th edition, 2001.

[27] CADNA: Control of Accuracy and Debugging for Numerical Applications. Université Pierre et Maris Curie, Paris. `http://www.lip6.fr/cadna`.

[28] R.E. Caflisch. Monte Carlo and Quasi-Monte Carlo Methods. In *Acta Numerica*, volume 7, pages 1–49. Cambridge University Press, Cambridge, 1998.

[29] A.C. Callahan. Random rounding: some principles and applications. In *Proc. 1976 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 501–504, Philadelphia, PA, USA, April 1976.

[30] F. Chaitin-Chatelin. Le calcul qualitatif. comment donner un sens à des resultats faux ? Technical Report TR/PA/95/10, CERFACS, 1995. Available at `http://www.cerfacs.fr/algor/reports/1995/TR_PA_95_10.ps.gz`.

[31] F. Chaitin-Chatelin and V. Frayssé. *Lectures on Finite Precision Computations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[32] F. Chaitin-Chatelin and E. Traviesas. PRECISE, a toolbox for assessing the quality of numerical methods and software. In *16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Lausanne, Switzerland, August 2000.

[33] F. Chaitin-Chatelin and E. Traviesas. *PRECISE and the reliability of Numerical Software*. Handbook for Numerical Computation. SIAM, 2002.

[34] M. Charikhi, J.-M. Chesneaux, F. Jézéquel, F. Rico, and L. Villain. A dynamical computation of a multiple integral involved in the neutron star theory. In *Proc. SCAN2002 conference, Paris, France*, September 2002.

[35] C.-Y. Chen. *Adaptive numerische Quadratur und Kubatur mit automatischer Ergebnisverifikation*. PhD thesis, Universität Karlsruhe, 1998.

[36] J.-M. Chesneaux. *Etude théorique et implémentation en ADA de la méthode CESTAC*. PhD thesis, Université Pierre et Marie Curie, Paris, 1988.

[37] J.-M. Chesneaux. CADNA : An ADA tool for round-off error analysis and for numerical debugging. In *ADA in Aerospace*, Barcelona, Spain, December 1990.

[38] J.-M. Chesneaux. Study of the computing accuracy by using probabilistic approach. In C. Ullrich, editor, *Contribution to Computer Arithmetic and Self-Validating Numerical Methods*, pages 19–30, IMACS, New Brunswick, New Jersey, USA, 1990.

[39] J.-M. Chesneaux. CADNA : Un outil pour le contrôle des arrondis et le débogage numérique en ADA. *La lettre ADA*, 48–49:14–22, November–December 1991.

[40] J.-M. Chesneaux. Descriptif d'utilisation du logiciel CADNA_F. Technical Report 92-32, Laboratoire MASI, June 1992.

[41] J.-M. Chesneaux. The quality relations in scientific computing. *Num. Algo.*, 7:129–143, 1994.

[42] J.-M. Chesneaux. *L'arithmétique stochastique et le logiciel CADNA*. Habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris, 1995.

[43] J.-M. Chesneaux, L.-S. Didier, and F. Rico. Fixed CADNA library. In *Proc. 5th Real Numbers and Computers conference*, Lyon, France, September 2003.

[44] J.-M. Chesneaux and F. Jézéquel. Dynamical control of computations using the trapezoidal and Simpson's rules. *J. of Universal Computer Science*, 4(1):2–10, 1998.

[45] J.-M. Chesneaux and J. Vignes. Sur la robustesse de la méthode CESTAC. *C. R. Acad. Sci. Paris Sér. I Math.*, 307:855–860, 1988.

[46] R. Chotin-Avot. *Architectures matérielles pour l'Arithmétique Stochastique Discrète*. PhD thesis, Université P. et M. Curie, Paris, France, June 2003.

[47] A.E. Connell and R.M. Corless. An experimental interval arithmetic package in maple. *Interval Computations*, (2):120–134, 1993.

[48] S.D. Conte and C. de Boor. *Elementary numerical analysis.* McGraw-Hill, International Student edition, 1980.

[49] R. Cools. Advances in multidimensional integration. *J. Comput. Appl. Math.*, 149:1–12, 2002.

[50] R. Cools. Extrapolation and adaptivity in software for automatic numerical integration on a cube. *Num. Algo.*, 34(2-4):259–270, 2003.

[51] R. Cools and A. Haegemans. CUBPACK: progress report. In T. O. Espelid and A. Genz, editors, *Numerical Integration - recent developments, software and applications*, volume 357 of *NATO ASI Series C: Mathematics and Physical Sciences*, pages 1–24. Kluwer Academic, Dordrecht, 1992.

[52] R. Cools and A. Haegemans. Algorithm 824: CUBPACK: A package for automatic cubature; framework description. *ACM Trans. Math. Softw.*, 29(3):287–296, September 2003.

[53] R. Cools, D. Laurie, and L. Pluym. Algorithm 764: CUBPACK++: A C++ package for automatic two-dimensional cubature. *ACM Trans. Math. Softw.*, 23(1):1–15, March 1997.

[54] R. Cools and B. Maerten. A hybrid subdivision strategy for adaptative integration routines. *J. of Universal Computer Science*, 4(5):485–499, 1998.

[55] R. Cools and P. Rabinowitz. Monomial cubature rules since 'Stroud': a compilation. *J. Comput. Appl. Math.*, 48:309–326, 1993.

[56] G.F. Corliss. Computing narrow inclusions for definite integrals. In E. Kaucher, U. Kulisch, and C. Ullrich, editors, *Computer arithmetic, scientific computation and programming languages.* B.G. Teubner, Stuttgart, 1987.

[57] G.F. Corliss, Ch. Hu, R. Ramirez, R.B. Kearfott, and G.W. Walster. Proposed level-1 interval basic linear algebra software (INTBLAS) subpackage for the new BLAS. Technical report, Marquette University and Sun Microsystems, 1997.

[58] G.F. Corliss and L.R. Rall. Adaptive, self-validating numerical quadrature. *SIAM J. Sci. Stat. Comput.*, 8:831–847, 1987.

[59] M. Daumas and J.-M. Muller, editors. *Qualité des calculs sur ordinateur.* Masson, Paris, 1997.

[60] P.J. Davis and P. Rabinowitz. *Methods of numerical integration.* Academic Press, London, 1984.

[61] D. Defour, G. Hanrot, V. Lefèvre, J.-M. Muller, N. Revol, and P. Zimmermann. Proposal for a standardization of mathematical function implementation in floating-point arithmetic. *Num. Algo.*, 37(1–4):367–375, December 2004.

[62] D. Defour and J.-M. Muller. *Evaluation des fonctions élémentaires*, volume 13 (4-5) of *Réseaux et systèmes répartis - calculateurs parallèles.* Editions Hermes, 2001.

[63] P. Van Dooren and L. De Ridder. An adaptative algorithm for numerical integration over an n-dimensional cube. *J. Comput. Appl. Math.*, 2:207–217, 1976.

[64] I. Ekeland. *Le calcul, l'imprévu.* Seuil, 1984.

[65] H. Engels. *Numerical quadrature and cubature.* Academic Press, 1980.

[66] J. Erhel, Y. Jégou, and B. Philippe. AQUARELS: A problem-solving environment for numerical quality. In *13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, Ireland, July 1991.

[67] J. Erhel, B. Philippe, N. Mallejac, and J.-C. Paoletti. AQUARELS: a problem-solving environment for numerical quality - review of the project and first use on realistic problems. In *ICIAM*, 1995.

[68] A. Feldstein and P. Turner. Overflow, underflow and severe loss of significance in floating point addition and subtraction. *IMA J. Numer. Anal.*, 6:241–251, 1986.

[69] G.E. Forsythe. Round-off errors in numerical integration on automatic machinery. preliminary report. *Bull. AMS*, 56(61), 1950.

[70] G.E. Forsythe. Note on rounding-off errors. *Math. Rev.*, 12(208), 1951.

[71] G.E. Forsythe. Reprint of a note on rounding-off errors. *SIAM Review*, 1(1):66–67, 1959. Originally written June 1950 at the National Bureau of Standards, Los Angeles, CA, and abstracted in [70].

[72] P. François. *Contribution à l'étude de méthodes de contrôle automatique de l'erreur d'arrondi : la méthodologie SCALP.* PhD thesis, Institut National Polytechnique de Grenoble, 1989.

[73] P. François and J.-M. Muller. The SCALP perturbation method. In *13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, Ireland, July 1991.

[74] A.C. Genz and A.A. Malik. An adaptative algorithm for numerical integration over an n-dimensional rectangular region. *J. Comput. Appl. Math.*, 6:295–302, 1980.

[75] W.J. Givens. Numerical computation of the characteristic values of a real symmetric matrix. Technical Report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge, TN, USA, 1954.

[76] M. Grimmer. Interval Arithmetic in Maple with intpakX. In *Proceedings in Applied Mathematics and Mechanics, GAMM-Conference, Augsburg, Germany, 2002*, volume 2, pages 442–443. Wiley-InterScience, 2003.

[77] P.C. Hammer and A.W. Wymore. Numerical evaluation of multiple integrals I. *MTAC*, 11:59–67, 1957.

[78] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz. *Numerical toolbox for verified computing, I.* Springer: Berlin, Heidelberg, New York, 1993.

[79] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz. *C++ Toolbox for Verified Computing.* Springer Verlag, Berlin, Heidelberg, 1995.

[80] G.I. Hargreaves. Interval analysis in MATLAB. Numerical Analysis Report No. 416, Manchester Centre for Computational Mathematics, University of Manchester, December 2002. Available at `http://www.maths.man.ac.uk/~nareports/narep416.pdf`.

[81] W. Harrison. Total energies in the tight-binding theory. *Phys. Rev.*, B23:5230–5245, 1981.

[82] N.J. Higham. *Accuracy and stability of numerical algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[83] O. Holzmann, B. Lang, and H. Schütt. Newton's constant of gravitation and verified numerical quadrature. *Reliab. Comput.*, 2(3):229–239, 1996.

[84] T.E. Hull and J.R. Swenson. Tests of probabilistic models for propagation of roundoff errors. *Comm. ACM*, 9(2):108–113, February 1966.

[85] IBM. ACRITH-XSC: IBM High accuracy arithmetic - Extended Scientific Computation. Technical report PI 881, Boblinger, IBM Deutschland GmbH, 1990.

[86] IEEE Computer Society, New York. *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985*, 1985. Reprinted in SIGPLAN Notices, 22(2):9-25, 1987.

[87] M. Iri. History of automatic differentiation and rounding error estimation. In A. Griewank and G.F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation and Application*, pages 3–16. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991.

[88] F. Jézéquel. Propagation de l'erreur d'arrondi lors de la résolution de l'équation des ondes. In *Proc. 2nd Real Numbers and Computers conference*, pages 215–231, Marseille, France, 1996.

[89] F. Jézéquel. Round-off error propagation in the solution of the heat equation by finite differences. *J. of Universal Computer Science*, 1(7):465–479, 1995.

[90] F. Jézéquel. A validated parallel across time and space solution of the heat transfer equation. *Applied Numerical Mathematics*, 31:65–79, 1999.

[91] F. Jézéquel. Dynamical control of converging sequences computation. *Applied Numerical Mathematics*, 50(2):147–164, 2004.

[92] F. Jézéquel, M. Charikhi, and J.-M. Chesneaux. Dynamical control of computations of multiple integrals. In *Proc. SCAN2002 conference, Paris, France*, September 2002.

[93] F. Jézéquel and J.-M. Chesneaux. Etude de la stabilité numérique du code OPA-version 8.0. Note technique du pôle de modélisation de l'IPSL, Institut Pierre Simon Laplace, March 1998. Available at `http://www.ipsl.jussieu.fr/modelisation/liste-notes-tech.html`.

[94] F. Jézéquel and J.-M. Chesneaux. For reliable and powerful scientific computations. In Krämer and Wolff von Gudenberg, editors, *Scientific Computing, Validated Numerics, Interval Methods*, pages 367–378. Kluwer Academic/Plenum Publishers, 2001.

[95] F. Jézéquel and J.-M. Chesneaux. Computation of an infinite integral using Romberg's method. *Num. Algo.*, 36(3):265–283, July 2004.

[96] R. Kearfott, M. Dawande, K. Du, and C. Hu. Algorithm 737: Intlib: A portable Fortran 77 interval standard function. *ACM Trans. Math. Softw.*, 20, 1992.

[97] R. Kelch. Numerical quadrature by extrapolation with automatic result verification. In *[2]*, pages 143–186.

[98] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and Ch. Ullrich, editors. *PASCAL-XSC, language reference with examples*. Springer: Berlin, Heidelberg, New York, 1991.

[99] R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo, and M. Rauch. *C-XSC, A C++ Class library for extended scientific computing*. Springer-Verlag, 1993.

[100] J. Kouatchou. Finite differences and collocation methods for the two dimensional heat equation. *Numerical Methods for PDEs*, 17:54–63, 2001.

[101] J. Kouatchou. Parallel implementation of a high-order implicit collocation method for the heat equation. *Math. Comput. Simulation*, 54(6):509–519, 2001.

[102] J. Kouatchou. Comparison of time and spatial collocation methods for the heat equation. *J. of Computational and Applied Mathematics*, 150(1):129–141, 2003.

[103] W. Krämer. Multiple-precision computations with result verification. In *[2]*, pages 325–356.

[104] W. Krämer and I. Geulig. Interval Calculus in Maple - The Extension intpakX to the Package intpak of the Share-Library. Preprint 2001/2, Universität Wuppertal, 2001.

[105] W. Krämer and W. Hofschuster. (The Future of) C-XSC. In *Validated Computing*. SIAM Workshop, Toronto, Canada, May 2002.

[106] W. Krämer, U. Kulisch, and R. Lohner. *Numerical toolbox for verified computing, II.* Springer: Berlin, Heidelberg, New York, 1999.

[107] W. Krämer and S. Wedner. Two adaptative Gauss-Legendre type algorithms for the verified computation of definite integrals. *Reliable Computing*, 2(3):241–253, 1996.

[108] U.W. Kulisch. *Advanced Arithmetic for the Digital Computer.* Springer-Verlag, Wien, 2002.

[109] U.W. Kulisch and W.M. Miranker, editors. *Computer Arithmetic in Theory and Pratice.* Academic Press, New York, 1981.

[110] J.-L. Lamotte. A tool for the numerical validation of parallel code programming with Fortran 77 and PVM. In *Proc. 2nd european conference EuroPVM*, pages 173–178, Lyon, France, 1995. Hermes.

[111] J.-L. Lamotte. On the visualization of the precision of validation software. In *SCAN2000, 9th International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics, Karlsruhe, Germany.* GAMM-IMACS, September 2000.

[112] J.-L. Lamotte. Parallelization of the CESTAC method on shared memory and distributed memory computers. In *10th International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics, SCAN2002, Paris, France.* GAMM-IMACS, September 2002.

[113] J.-L. Lamotte. *Vers une chaine de validation des logiciels numériques à l'aide de méthodes probabilistes.* Habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris, November 2004.

[114] J.-L. Lamotte and Y. Epelboin. Study of the numerical stability of an X-RAY diffraction model. In *Computational Engineering in Systems Applications, CESA'98 IMACS Multiconference, Nabeul-Hammamet, Tunisia*, April 1998.

[115] J.-L. Lamotte and D. Martins. First parallel implementation of the CESTAC method with self validation. Mathematical Modelling and Scientific Computations, minisymposium within the 33rd Spring Conference of the Union of the Mathematicians in Bulgaria, Borovets, Bulgaria, April 2004.

[116] B. Lang. Verified quadrature in determining Newton's constant of gravitation. *J. Univ. Comput. Sci.*, 4(1):16–24, 1998.

[117] B. Lang. Derivative-based subdivision in multi-dimensional verified Gaussian quadrature. In G. Alefeld, J. Rohn, S. Rump, and T. Yamamoto, editors, *Symbolic Algebraic Methods and Verification Methods*, pages 145–152, Wien, 2001. Springer-Verlag.

[118] P. Langlois. Automatic linear correction of rounding errors. *BIT*, 41(3):515–539, September 2001.

[119] P. Langlois. *Précision finie et méthodes automatiques*. Habilitation à diriger des recherches, Université Claude Bernard Lyon 1, July 2001.

[120] P. Langlois. *Une méthode de correction automatique*, volume 13 (4-5) of *Réseaux et systèmes répartis - calculateurs parallèles*, pages 505–522. Editions Hermes, 2001.

[121] P. Langlois. Improving the CENA method with the IA'64, Itanium(TM) processor. In *Proc. SCAN2002 conference, Paris, France*, September 2002.

[122] P. Langlois and F. Nativel. Reduction and bounding of the rounding error in floating point arithmetic. *C.R. Acad. Sci. Paris, Série 1*, 327:781–786, 1998.

[123] P. Langlois and F. Nativel. When automatic linear correction of rounding errors is exact. *C.R. Acad. Sci. Paris, Série 1*, 328:543–548, 1999. Erratum in 328:829, 1999.

[124] P. Langlois and N. Revol. Validating polynomial numerical computation with complementary automatic methods. Research Report 2001-18, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, France, May 2001.

[125] V. Lefèvre, J.-M. Muller, and A. Tisserand. Toward correctly rounded transcendentals. *IEEE Transactions on Computers*, 47(11), November 1998.

[126] S. Linnainmaa. Towards accurate statistical estimation of rounding errors in floating-point compuations. *BIT*, 15:165–173, 1975.

[127] S. Linnainmaa. Error linearization as an effective tool for experimental analysis of the numerical stability of algorithms. *BIT*, 23:346–359, 1983.

[128] R. Lohner. Enclosures for the solutions of ordinary initial and boundary value problems. In J.R. Cash and I. Gladwell, editors, *Computational Ordinary Differential Equations*, pages 425–435, Oxford, 1992. Clarendon Press.

[129] J.N. Lyness. Symmetric integration rules for hypercubes. I, II, III. *Math. Comp.*, 19:260–276, 394–407, 625–637, 1965.

[130] J.N. Lyness. An error functional expansion for N-dimensional quadrature with an integrand function singular at a point. *Math. Comp.*, 30:1–23, 1976.

[131] J.N. Lyness and R. Cools. A survey of numerical cubature over triangles. In W. Gautschi, editor, *Mathematics of computation 1943-1993: A half-century of computational mathematics, Proceedings of Symposia in Applied Mathematics*, volume 48, pages 127–150. American Mathematical Society, Providence, Rhode Island, 1994.

[132] J.N. Lyness and E. de Doncker. Quadrature error expansions. II. The full corner singularity. *Numer. Math.*, 64:355–370, 1993.

[133] J.N. Lyness and B.J.J. McHugh. On the remainder term in the N-dimensional Euler-Maclaurin expansion. *Numer. Math.*, 15:333–344, 1970.

[134] J.N. Lyness and K.K. Puri. The Euler-Maclaurin expansion for the simplex. *Mathematics of computation*, 27(122):273–293, 1973.

[135] G. Madec, P. Delecluse, M. Imbard, and C. Lévy. OPA 8.1 Ocean General Circulation Model reference manual. Note du Pôle de modélisation 11, Institut Pierre-Simon Laplace, 1998.

[136] Maple. Waterloo Maple Inc, Waterloo, ON, Canada. `http://www.maplesoft.com`.

[137] S. Markov. Extended interval arithmetic involving infinite intervals. *Mathematica Balkanica*, 6:269–304, 1992.

[138] S. Markov. On the presentation of ranges of monotone functions using interval arithmetic. *Interval Computations*, 4:19–31, 1992.

[139] S. Markov and R. Alt. Stochastic arithmetic: addition and multiplication by scalars. *Applied Numerical Mathematics*, 50(3–4):475–488, 2004.

[140] J.H. Mathews. *Numerical methods for mathematics, science and engineering.* Prentice-Hall, 2nd edition, 1992.

[141] W. Miller. Software for roundoff analysis. *ACM Trans. Math. Softw.*, 1:108–128, 1975.

[142] M. Montagnac. *Contrôle dynamique d'algorithmes itératifs de résolution de systèmes linéaires.* PhD thesis, Université Pierre et Marie Curie, Paris, 1999.

[143] W.J. Morokoff and R.E. Caflisch. Quasi-Monte Carlo Integration. *J. Comp. Phys.*, 122(2):218–230, 1995.

[144] J.-M. Muller. *Arithmétique des Ordinateurs.* Masson, 1989.

[145] J.-M. Muller. *Elementary Functions, Algorithms and Implementation.* Birkhauser, Boston, 1997.

[146] V. Ménissier. *Arithmétique exacte : conception, algorithmique et performances d'une implantation informatique en précision arbitraire.* PhD thesis, Université Pierre et Marie Curie, Paris, 1994.

[147] A. Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confidence regions. *Computing (Suppl.)*, 9:175–190, 1993.

[148] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* SIAM, Philadelphia, 1992.

[149] D.S. Parker. Monte Carlo Arithmetic: Exploiting randomness in floating-point arithmetic. Technical Report 970002, UCLA Computer Science Dept., 1997. Available at `http://www.cs.ucla.edu/~stott/mca/CSD-970002.ps.gz`.

[150] D.S. Parker, P.R. Eggert, and B. Pierce. Monte Carlo Arithmetic: A framework for the statistical analysis of roundoff error. Technical Report 970014, UCLA Computer Science Dept., 1997. Available at `http://www.cs.ucla.edu/~stott/mca/CSD-970002.ps.gz`.

[151] D.S. Parker, B. Price, and P. Eggert. Monte Carlo Arithmetic: How to gamble with floating point and win. *Computing in Science and Engineering*, pages 58–68, July–August 2000.

[152] M. Pichat and J. Vignes. Validité des résultats numériques dans les processus à comportement chaotique. un outil d'évaluation : le logiciel CADNA. *C.R. Acad. Sci. Paris*, pages 681–688, 1996.

[153] M. Pichat and J. Vignes. The numerical study of unstable fixed points in a chaotic dynamical system. In *15th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, pages 341–346, Berlin, Germany, August 1997.

[154] M. Pichat and J. Vignes. The numerical study of chaotic systems - future and past. In *16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Lausanne, Switzerland, August 2000.

[155] R. Piessens, E. de Doncker-Kapenga, C.W. Überhuber, and D.K. Kahaner. *QUADPACK, A Subroutine Package for Automatic Integration.* Springer Series in Computational Mathematics. Springer Verlag, 1983.

[156] PRECISE: PRecision Estimation and Control in Scientific and Engineering computing. `http://www.cerfacs.fr/algor/Softs/PRECISE/`.

[157] Problem 95-7. SIAM Review, V 38, N 2, page 324, 1996.

[158] L. Prylli and B. Tourancheau. BIP: a new protocol designed for high performance networking on Myrinet. In *Workshop PC-NOW, IPPS/SPDP98*, Orlando, USA, 1998.

[159] L.B. Rall. Differentiation and generating Taylor coefficients in PASCAL-SC. In *[109]*.

[160] R.E. Moore. *Interval Analysis*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1966.

[161] N. Revol and F. Rouillier. MPFI: Multiple Precision Floating-point Interval library. `http://perso.ens-lyon.fr/nathalie.revol/software.html`.

[162] S.M. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):534–554, 1999.

[163] S.M. Rump. INTLAB - INTerval LABoratory. In T. Csendes, editor, *Developments in reliable computing. SCAN-98 conference, 8th international symposium on Scientific computing, computer arithmetic and validated numerics. Budapest, Hungary, 1998*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[164] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[165] A. Sidi. Euler-Maclaurin expansions for integrals over triangles and squares of functions having algebraic/logarithmic singularities along an edge. *J. Approx. Theory*, 39:39–53, 1983.

[166] S.Markov. Some applications of extended interval arithmetic to interval iterations. *Computing*, Supl.2:69–84, 1980.

[167] J. Stewart. *Calculus: Early Transcendentals*. Brooks/Cole Publishing Company, 5th edition, 2002.

[168] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12 of *Texts in applied mathematics*. Springer, 3rd edition, 2002.

[169] U. Storck. Numerical integration in two dimensions with automatic result verification. In *[2]*, pages 187–224.

[170] U. Storck. Verified calculation of the nodes and weights for gaussian quadrature formulas. *Interval Computation*, 4:114–124, 1993.

[171] U. Storck. An adaptative numerical integration algorithm with automatic result verification for definite integrals. *Computing*, 65:271–280, 2000.

[172] M. Tienari. A statistical model of roundoff error for varying length floating-point arithmetic. *BIT*, 10:355–365, 1970.

[173] R. van der Pas and B. Walster. Interval arithmetic support in the Sun Fortran compiler. In *NSF workshop on Reliable Engineering Computing, Savannah, Georgia, USA*, September 2004.

[174] P. Verlinden and A. Haegemans. An error expansion for cubature with an integrand with homogeneous boundary singularities. *Numer. Math.*, 65:383–406, 1993.

[175] P. Verlinden, D.M. Potts, and J.N. Lyness. Error expansions for multidimensional trapezoidal rules with Sidi transformations. *Num. Algo.*, 16(3-4):321–347, 1997.

[176] J. Vignes. New methods for evaluating the validity of the results of mathematical computations. *Math. Comput. Simulation*, 20:227–249, 1978.

[177] J. Vignes. Zéro mathématique et zéro informatique. *C. R. Acad. Sci. Paris Sér. I Math.*, 303:997–1000, 1986. also: La Vie des Sciences, 4 (1) 1-13, 1987.

[178] J. Vignes. Estimation de la précision des résultats de logiciels numériques. *La Vie des Sciences*, 7(2):93–145, 1990.

[179] J. Vignes. A stochastic arithmetic for reliable scientific computation. *Math. Comput. Simulation*, 35:233–261, 1993.

[180] J. Vignes. A stochastic approach to the analysis of round-off error propagation. A survey of the CESTAC method. In *Proc. 2nd Real Numbers and Computers conference*, pages 233–251, Marseille, France, 1996.

[181] J. Vignes. Discrete stochastic arithmetic for validating results of numerical software. *Num. Algo.*, 37(1–4):377–390, December 2004.

[182] J. Vignes and R. Alt. An efficient stochastic method for round-off error analysis. In W.L. Miranker and R.A. Toupin, editors, *Accurate scientific computations*, volume 235 of *Lecture Notes in Computer Science*, pages 183–205, Symposium Bad Neuenahr, Germany, March 1985. Springer Verlag.

[183] J. Vignes and M. La Porte. Error analysis in computing. In *Information Processing 1974*, pages 610–614. North-Holland, 1974.

[184] L. Villain. *Instabilités hydrodynamiques relativistes et processus URCA hors équilibre dans les étoiles à neutrons en rotation.* PhD thesis, Université Paris 7, Paris, 2002.

[185] S. Wedner. *Verifizierte Bestimmung singulärer Integrale - Quadratur und Kubatur.* PhD thesis, Universität Karlsruhe, 2000.

[186] J.H. Wilkinson. Rounding errors in algebraic processes. In *Notes on Applied Science*, number 32. Her Majesty's Stationery Office, London, 1963. Also published by Prentice-Hall, Englewood Cliffs, New Jersey, USA. Reprinted by Dover, New York, 1994.

[187] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, 1965.

[188] J.H. Wilkinson. Modern error analysis. *SIAM review*, 13(4):548–568, 1971.

[189] J.H. Wilkinson. Error analysis revisited. *IMA Bulletin*, 22(11-12):192–200, 1986.

Chapitre 8

Principales publications

Nous présentons dans ce chapitre les documents suivants :

1. F. Jézéquel,
   Round-off error propagation in the solution of the heat equation by finite differences,
   J. of Universal Computer Science, 1(7): 465-479, 1995.
2. F. Jézéquel,
   Propagation de l'erreur d'arrondi lors de la résolution de l'équation des ondes,
   Actes du congrès Real Numbers and Computers, 215-231, Marseille, France, 1996.
3. F. Jézéquel,
   A validated parallel across time and space solution of the heat transfer equation,
   Applied Numerical Mathematics 31: 65-79, 1999.
4. F. Jézéquel, J.-M. Chesneaux,
   Étude de la stabilité numérique du code OPA-version 8.0,
   Note technique du pôle de modélisation de l'IPSL (Institut Pierre Simon Laplace), 1998.
5. F. Jézéquel, J.-M. Chesneaux,
   For reliable and powerful scientific computations,
   Scientific Computing, Validated Numerics, Interval Methods, Krämer and Wolff von Gudenberg ed., Kluwer Academic/Plenum Publishers, 367-378, 2001.
6. J.-M. Chesneaux, F. Jézéquel,
   Dynamical control of computations using the trapezoidal and Simpson's rules,
   J. of Universal Computer Science, 4(1): 2-10, 1998.
7. F. Jézéquel,
   Dynamical control of converging sequences computation,
   Applied Numerical Mathematics, 50(2): 147-164, 2004.
8. F. Jézéquel, J.-M. Chesneaux,
   Computation of an infinite integral using Romberg's method,
   Numerical Algorithms, 36(3): 265-283, 2004.
9. F. Jézéquel, F. Rico, J.-M. Chesneaux, M. Charikhi,
   Reliable computation of a multiple integral involved in the neutron star theory,
   article soumis à la revue "Mathematics and Computers in Simulation".

8.1 Sur la propagation d'erreur d'arrondi lors de la résolution de l'équation de la chaleur

F. Jézéquel,
Round-off error propagation in the solution of the heat equation by finite differences,
J. of Universal Computer Science, 1(7): 465-479, 1995.

8.2 Sur la propagation d'erreur d'arrondi lors de la résolution de l'équation des ondes

F. Jézéquel,
Propagation de l'erreur d'arrondi lors de la résolution de l'équation des ondes,
Actes du congrès Real Numbers and Computers, 215-231, Marseille, France, 1996.

8.3 Sur la résolution parallèle en temps et en espace de l'équation de la chaleur

F. Jézéquel,
A validated parallel across time and space solution of the heat transfer equation,
Applied Numerical Mathematics 31: 65-79, 1999.

8.4 Sur la stabilité numérique du code OPA

F. Jézéquel, J.-M. Chesneaux, *Étude de la stabilité numérique du code OPA-version 8.0*, Note technique du pôle de modélisation de l'IPSL (Institut Pierre Simon Laplace), 1998.

8.5 Sur l'estimation d'erreur d'arrondi sur calculateur vectoriel

F. Jézéquel, J.-M. Chesneaux,
For reliable and powerful scientific computations,
Scientific Computing, Validated Numerics, Interval Methods, Krämer and Wolff von Gudenberg ed., Kluwer Academic/Plenum Publishers, 367-378, 2001.

8.6 Sur le contrôle dynamique des méthodes des trapèzes et de Simpson

J.-M. Chesneaux, F. Jézéquel,
Dynamical control of computations using the trapezoidal and Simpson's rules,
J. of Universal Computer Science, 4(1): 2-10, 1998.

8.7 Sur le contrôle dynamique de calculs de suites

F. Jézéquel,
Dynamical control of converging sequences computation,
Applied Numerical Mathematics, 50(2): 147-164, 2004.

8.8 Sur le calcul fiable d'une intégrale impropre issue de la cristallographie

F. Jézéquel, J.-M. Chesneaux,
Computation of an infinite integral using Romberg's method,
Numerical Algorithms, 36(3): 265-283, 2004.

8.9 Sur le calcul fiable d'une intégrale multiple issue de la théorie des étoiles à neutrons

F. Jézéquel, F. Rico, J.-M. Chesneaux, M. Charikhi,
Reliable computation of a multiple integral involved in the neutron star theory,
article soumis à la revue "Mathematics and Computers in Simulation".