Rapport d'avancement thèse ACSEL

Comité de Suivi Individuel 09/2018

Sous la direction de David Defour, MCF HDR 27e et Bijan Mohammadi, Pr, 26e

Christophe Pont

3 octobre 2018

Table des matières

1 Présentation

1.1 Présentation personnelle

Après l'obtention d'un Bac S spé maths (mention bien) à Toulouse, j'ai intégré un DUT Informatique, que j'ai obtenu avec la mention Bien. Mon cursus s'est prolongé à Polytech Montpellier, où j'y ai obtenu mon diplôme

d'ingénieur en Informatique et Gestion (mention Bien). Durant cette formation, j'ai pu réaliser un stage anglophone dans un laboratoire de bioinformatique à l'université d'Ottawa, au Canada. Ce premier contact avec la recherche m'a incité à démarrer un doctorat, en octobre 2017, après deux ans en tant qu'ingénieur logiciel dans une grande entreprise de service informatique (SSII). L'idée était d'harmoniser mon orientation avec mon attrait pour la science, en particulier l'informatique et les mathématiques. C'est pourquoi je me suis dirigé vers le laboratoire LAMPS, et ai démarré mon doctorat sous la direction de David Defour, HDR, 27e afin de démarrer une thèse mêlant ma compétence informatique avec mon attrait pour les mathématiques : le tout, dans un contexte écologique qui me tient à cœur. J'aspire à continuer ma carrière dans la recherche, qui sait peut-être en tant que maître de conférence.

1.2 Contexte

Le contexte dans lequel se place mon sujet de thèse est double. D'un côté, on s'intéresse à l'**arithmétique flottante**.
Les nombres flottants peuvent se définir comme une discrétisation des nombres réels. Le Standard IEEE 754 [13] définit 3 entiers (s, e, m) pour représenter nombres flottants (en l'occurence s étant le bit de signe, e l'exposant et m la mantisse). Il fournit aussi des **formats**, qui définissent la façon dont chaque nombre flottant est **encodé**. On peut citer le format simple précision, le format double précision etc.
Prenons l'exemple d'une telle discrétisation (Figure 1) selon les paramètres suivants : $\beta = 2, p = 3, e_{min} = -1, e_{max} = 2$, avec β la base, p la précision (soit le nombre de bits de la mantisse), e_{min} l'exposant minimal et e_{max} l'exposant maximal. Sur l'exemple, on voit bien que $1,25$ est représentable $(1.01 * 2^0)$, de même que 3 $(1.10 * 2^1)$. Bien que la somme de ces deux réels donne $3,25$, la somme flottante de ces deux nombres flottants donne elle 3 (après arrondi), car $3,25$ n'est pas représentable exactement dans ces conditions.

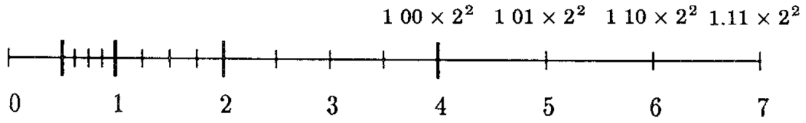

Figure 1. Normalized numbers when $\beta = 2$, $p = 3$, $e_{\min} = -1$, $e_{\max} = 2$.

FIGURE 1 – Exemple de discretisation des nombres réels en nombres flottants. Source : [5]

Malgré la définition d'un standard, l'arithmétique flottante est soumise à de multiples erreurs. On pense notamment aux **erreurs d'arrondi**, comme dans le cas cité ci-dessus où $3,25$ a dû être arrondi à 3, entraînant la *perte* de $0,25$. Mais on peut aussi citer les **erreurs de cancellation**, qui surviennent lorsque l'on soustrait deux grandes valeurs très proches l'une de l'autre, révélant la perte de nombre significatifs lors de précédents calculs. Hors, lorsque l'on simule un modèle qui comprend de nombreux calculs (super calculateurs, vastes simulations à exécution longue), les différentes erreurs s'accumulent, se combinent, et finalement affectent le résultat.

Le deuxième volet de ma thèse porte sur ce domaine, celui de la **recherche d'erreurs numériques dans les programmes**. Le cas d'étude de ma thèse porte sur les modèles et simulations de **Gladys** [1]. Gladys est un réseau international de littoralistes en méditerranée, regroupant de nombreux chercheurs dans le domaine du littoral. Ils ont établi de nombreux modèles pour décrire les phénomènes afférents à leur activité, notamment le phénomène **d'évolution du trait de côte**. L'implémentation informatique de ce modèle sera le sujet d'étude de ma thèse (un code volumineux en C et Fortran). Il prend des données en entrée et réalise sa simulation. C'est ici que viendra se placer mon travail. Le principe recherché est de **compiler** le programme, puis d'en produire une analyse visant à suivre et quantifier les erreurs liées à l'arithmétique flottante.

1.3 État de l'art

On peut distinguer deux catégories d'initiatives existantes dans le domaine : les démarches visant à *limiter* l'effet des erreurs (on peut citer les sommations compensées [3], Salsa [1]...), et celles visant à quantifier et mesurer les erreurs. Ma thèse fait écho à cette deuxième approche. La riche littérature existante fait état de nombreux outils pouvant réaliser cette analyse. Chacun dispose cependant de différentes entrées (programmes, modèles, binaires...), et différentes contraintes. On s'intéresse ici à deux approches probabilistes en particulier : la méthode CESTAC [2], et l'arithmétique de Monte Carlo [3]. La méthode CESTAC [10] utilise un grand nombre de tirages pour approximer efficacement le nombre de bits significatifs du résultat. DSA [4] [12], basée sur CESTAC, redéfinit quant à elle les opérateurs relationnels pour effectuer chaque opération N fois, en changeant aléatoirement le mode d'arrondi (et définit aussi les notions de nombre stochastique et de zéro stochastique). Enfin, CADNA [5] [8] implémente DSA pour $N = 3$, où les deux premiers modes d'arrondis sont choisis aléatoirement, et où le

1. `http://www.gladys-littoral.org`
2. Controle et Estimation Stochastique des Arrondis de Calculs
3. ou MCA (Monte Carlo Arithmetic)
4. Discrete Stochastic Arithmetic
5. Control of Accuracy and Debugging for Numerical Application

troisième est différent du second.
Parallèlement, MCA [9] permet une analyse de sensibilité, en utilisant notamment un *arrondi randomisé* pour ajouter du bruit aux nombres flottants, à l'aide d'une variable aléatoire uniformément distribuée, afin de quantifier et compenser les erreurs d'arrondi notamment. Finalement, Verificarlo [4] est une implémentation de MCA utilisant LLVM [6] pour la portabilité. On citera aussi FPANR [7] [2], encodage proposé par David Defour, qui comme on va le voir encode astucieusement la précision dans la mantisse.

2 Travail réalisé

2.1 Arrondi randomisé

Nous avons utilisé l'arrondi randomisé, inspiré du *round_ random_nearness* de MCA. La motivation est simple : à la fois tracer les erreurs en utilisant notamment l'écart type des tirages, ainsi que *compenser statistiquements* les erreurs d'arrondis à l'aide du grand nombre de tirage. Le principe est le suivant : au lieu d'arrondi au flottant le plus proche, comme le préconiserait l'arrondi au plus près, mode d'arrondi standard et largement utilisé, l'arrondi randomisé va introduire une variable aléatoire uniformément distribuée pour déterminer le sens de l'arrondi (vers le *haut* ou vers le *bas*). La probabilité d'arrondir au flottant supérieur (resp. inférieur) à la valeur réelle que l'on souhaite arrondir **est inversement proportionnelle à la distance de ce réel au flottant supérieur (resp. inférieur)**. En somme, là où l'arrondi au plus près arrondit toujours $0,49$ à 0, l'arrondi randomisé arrondira $0,49$ à 0 avec une probabilité $P(\lfloor x \rfloor) = 0,51$ et arrondira à 1 avec une probabilité $P(\lfloor x \rfloor + \epsilon) = 0,49$, ϵ correspondant à l'*epsilon machine*, soit l'erreur relative maximale due à une erreur d'arrondi pour cette base et cette précision. En résumé, l'arrondi randomisé peut réduire les problèmes d'accumulation d'erreurs d'arrondis, mais de manière *statistique* (et non locale).

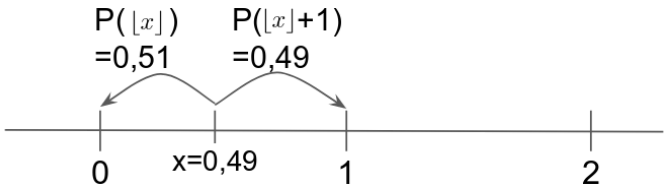

FIGURE 2 – Exemple d'arrondi randomisé

On s'intéressera notamment à l'article *Deep Learning with Limited Numerical Precision* [7], où leur démarche est d'entraîner des réseaux de neu-

6. `https://llvm.org/`
7. Floating-Point Adaptive Noise Reduction

rones avec des formats restreints (en fixed-point), à faible précision, pour observer la sensibilité de l'apprentissage au mode d'arrondi. Ils observent que lorsque la précision diminue trop, un phénomène de divergence dans l'apprentissage se dessine. Pour compenser, ils introduisent un arrondi randomisé. Dans ce cas, ils notent qu'en fonction du mode d'arrondi, on peut observer une divergence du gradient ou une convergence (en faveur de l'arrondi randomisé). L'idée est donc de tester cet arrondi randomisé dans d'autres cas, et finalement reproduire leurs résultats, les approfondir et les expliquer. Une piste intéressante pour expliquer cette convergence est que l'introduction de bruit s'avère positive dans l'apprentissage de CNN : on l'ajoute donc volontairement lors de l'entraînement. De la même façon, il serait intéressant de voir dans quelle mesure le bruit induit par l'arrondi stochastique pourrait avoir un impact positif sur l'apprentissage.
Avant cela, nous avons souhaité tester l'arrondi randomisé sur des procédés connus pour être numériquement instables. En l'ocurrence, le gradient conjugué, et l'attracteur de Lorenz. Le gradient conjugué est un procédé itératif permettant de résoudre des systèmes d'équations linéaires [11]. L'attracteur de Lorenz, procédé itératif utilisé notamment en météorologie, s'avère d'autnat plus sensible à l'accumulation d'erreurs dans les calculs flottants, donc potentiellement au mode d'arrondi utilisé.

L'implémentation de ces deux procédés a été assez complexe, puisqu'il a fallu utiliser un mode d'arrondi codé *à la main* dans chaque opération. Chaque programme est ensuite exécuté une fois avec les modes d'arrondis *communs*, puis de multiples fois avec l'arrondi stochastique, dont on récolte finalement la moyenne ainsi que l'écart type des valeurs finales.

2.1.1 Lorenz

Les paramètres de l'attracteur de Lorenz utilisé sont $\sigma = 10$, $\rho = 28$ et $\beta = 8/3$. Aucun *fine-tuning* n'a été réalisé pour l'heure sur ces paramètres : ceux par défaut ont été gardés. On compare les valeurs testées à une exécution considérée *optimale*, réalisée avec une forte précision, et on obtient les résultats de la Figure 3.
Les axes en échelle logarithmique représentent en abscisse le temps (en nombre d'itérations) et en ordonnée la distance relative à l'optimal. Plus cette distance est grande, plus on s'éloigne de la valeur *correcte*.

On observe clairement les conséquences de l'accumulation d'erreur, car à mesure des itérations en perd de plus en plus en précision. CADNA semble fournir des résultats équivalents à ceux du mode d'arrondi standard (RNDN), dans l'ensemble plus précis que l'arrondi randomisé sur ce cas d'étude.

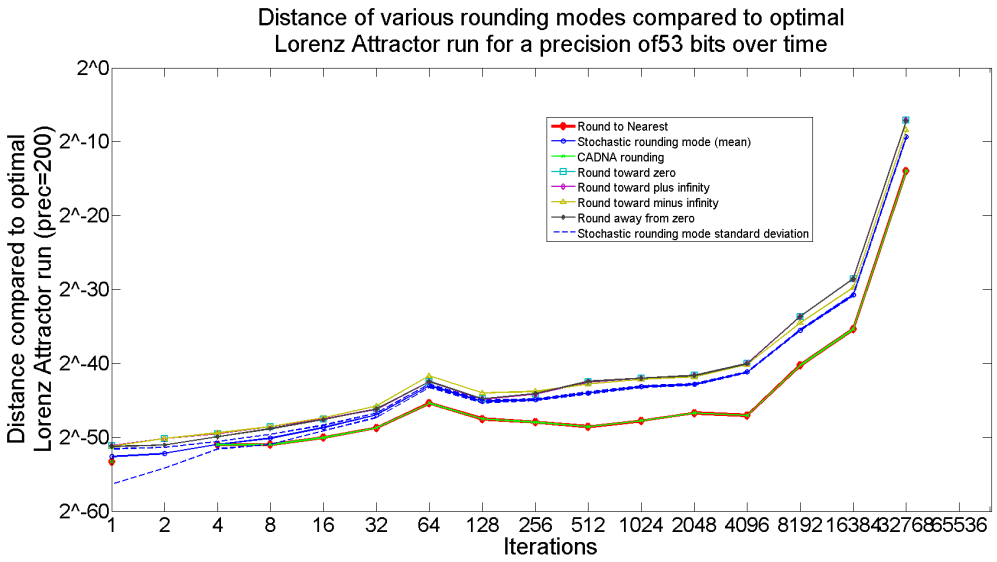

FIGURE 3 – Suivi de l'accumulation d'erreur en fonction du temps et du mode d'arrondi, à une précision de 53 bits (format double précision)

2.1.2 Gradient conjugué

Concernant le gradient conjugué, on s'est proposé de comparer les deux modes d'arrondi en fonction du nombre d'itérations, ainsi que de la précision des mantisses utilisées. Les résultats sont semblables à ceux obtenus avec Lorenz : l'arrondi au plus près semble là aussi plus efficace. En revanche, ce cas n'a pas été comparé à CADNA. Des courbes de ces résultats seront intégrées à ce rapport une fois générées de nouveau.

Avant de me lancer dans l'extension aux réseaux de neurones, j'ai décidé de réaliser l'implémentation de FPANR en juin dernier.

2.2 FPANR

FPANR est un format d'encodage particulier, présenté dans [2]. Le principe est d'encoder un flottant d'une manière permettant de **stocker** dans la mantisse le nombre de bits non significatifs de la valeur flottante elle-même. On a vu qu'au cours des calculs, un nombre flottant peut être amené à accumuler du bruit. La partie de sa mantisse située du côté de bits de poids faible devient alors non signifiante.
L'idée d'enregistrer le nombre de bits significatifs en parallèle du flottant trouve son origine dans *Significance arithmetic* [6] FPANR propose un mécanisme astucieux permettant de **marquer** la partie signifiante, à l'aide d'un *pattern* spécifique dans la mantisse ; l'incertitude est ainsi directement

représentée, sans surcoût mémoire. On conserve ainsi la compatibilité avec le standard IEEE754, ce qui s'avère utile pour l'allocation mémoire et l'implémentation *hardware*.
Le *coût* engendré est un motif (indiquant la non significance) à la fin du nombre flottant (1000....). Ce pattern viendra remplacer des bits qui ont perdu leur sens au fil des erreurs d'arrondis. Le risque survient lorsque l'on manipule un nombre dont la précision est maximum, où FPANR se doit de *tagger* le dernier bit, qui pourtant était correct. Le motif diminue donc la précision d'au plus 1 bit dans le pire des cas. Autre considération, les opérations entre nombres encodés FPANR ne peuvent se faire qu'après une reconversion de ceux-ci en flottants IEEE 754, ce qui peut entraîner des conséquences en terme de performance et de temps d'exécution.

Cette démarche s'intègre dans la démarche de ma thèse en ce sens que FPANR permet de suivre la précision au fil de l'exécution. On se propose d'en réaliser une intégration LLVM, afin de permettre une utilisation sans modification explicite des sources cible.

L'objectif dans le cadre de ma thèse est de rendre possible l'intégration de cet encodage dans n'importe quel programme. A cette fin, et plutôt que de redéfinir des passes LLVM "à la main", nous avons choisi de nous baser sur verificarlo (qui contient déjà ces passes).

L'expérience est un succès puisque nous avons déjà réussi à manipuler des nombres FPANR, et à les expérimenter sur quelques cas. Cependant, nous nous sommes heurtés à quelques difficultés. IO : FPANR nécessite une conversion explicite entre un flottant IEEE 754 et un flottant FPANR. Il est donc nécessaire de transformer toutes les entrées des programmes cible (y compris les constantes) pour pouvoir manipuler correctement les nombres dans le programme. Pour y parvenir, plusieurs solutions : coder à la main les conversions, à l'aide d'une librairie les contenant. Pratique à court terme, mais l'objectif est d'automatiser cela grâce à LLVM. On envisage aussi des options en utilisant des pragmas, mais c'est en cours de réflexion.

Il faudra aussi s'interroger sur la politique à adopter lorsque l'on se retrouve avec un nombre qui serait entièrement non significatif : le signale-t-on à l'utilisateur, si oui comment etc En dernier lieu, on utilise actuellement un développement en série de Taylor d'ordre 1 pour estimer l'évolution des bits significatifs sur certaines fonctions, or il faut s'assurer que les hypothèses sous tendues sur les dérivées se vérifient en pratique.

3 Publications en cours

Rédaction de mes résultats sur l'arrondi randomisé appliqué au gradient conjugué, à l'attracteur de Lorenz (et bientôt aux réseaux de neurones). Publication des bancs de tests réalisés sur la bibliothèque FPANR, sur les

SPEC notamment.

4 Planning prévisionnel

commencé à me familiariser avec les erreurs numériques sur des cas tels que la descente de gradient observer l'effet de l'arrondi randomisé

Confrontation verificarlo, cadna, qui pourront me servir de repères

Application aux modèles de prédiction d'érosion du littoral de GLADYS Récolte de statistiques sur les programmes par méthode de Monte Carlo

finalement, l'idée sera d'extraire l'information pertinente de ces statistiques, avec des méthodes de Data Mining (Random Forest) et de la délivrer à un format intelligible à l'utilisateur, via des indicateurs (voire des recommandations). Pour un plus large champ d'application ==> LLVM

(; et de le faire de manière assez générique pour que cela s'applique aux modèles de gladys - mais pas uniquement. (faire un outil qui pourra être applicable dans différents domaines ; on envisageait d'utiliser l'outil de compilation libre LLVM, pour réaliser des passes automatiques dans les implémentations des modèles afin d'en extraire de l'information par exemple) analyse descriptive)

analyser erreurs, (identifier). intéresser sensibilité arrondi randomisé

commencé à me familiariser avec les erreurs numériques (gradient, DNN) varier précision/mode d'arrondi, regarder comment cela impacte l'apprentissage

Confrontation automatic differentiation, verificarlo, cadna, qui sont d'autres outils dans le domaine, qui pourront me servir de repères

Application aux modèles de prédiction d'érosion du littoral de GLADYS Récolte de statistiques sur les programmes par méthode de Monte Carlo

finalement, l'idée sera d'extraire l'information pertinente de ces statistiques, avec des méthodes de Data Mining (Random Forest) et de la délivrer à un format intelligible à l'utilisateur, via des indicateurs (voire des recommandations). Pour un plus large champ d'application ==> LLVM

(; et de le faire de manière assez générique pour que cela s'applique aux modèles de gladys - mais pas uniquement. (faire un outil qui pourra être applicable dans différents domaines ; on envisageait d'utiliser l'outil de compilation libre LLVM, pour réaliser des passes automatiques dans les implémentations des modèles afin d'en extraire de l'information par exemple) analyse descriptive)

4.1 FPANR - SPEC

Avant de passer notre outil sur les codes de Gladys, on tient à les confronter à des jeux de données reconnus dans le domaine (et moins volumineux) : les SPEC. Passer notre outil FPANR sur les SPEC, afin de produire des

résultats renforçant la publication. Les spec sont des jeux de données standardisés qui seront un excellent “test” des capacités de la bibliothèque implémentée ==> conversions, il faut gérer les entrées sorties. Pour ça, deux solutions : soit via des fonctions à intégrer directement et manuellement dans le code (version simple pour avoir des résultats dans les spec). Solution idéale, détecter ces conversions dans LLVM, et les remplacer automatiquement (nécessite de s'intégrer à Clang pour créer un type particulier). D'autres solutions, notamment via des pragma, mais on en discute encore.

4.2 Arrondi randomisé

vu que on voit que FPANR peut vite perdre de la précision, on va éventuellement appliquer l'arrondi randomisé dans le cadre de FPANR

prolonger la descente de gradient en étudiant les réseaux de neurones

4.3 Gladys

l'objectif de ma thèse c'est de voir un vrai code. aidé par des experts qu'on a dans la région, qui sont compétents sur le sujet : les modèles et simulations de Gladys. De plus, les problèmes traités ont un réel impact sociétal (montée des eaux, écologie, immobilier, etc).

Une fois qu'on aura un outil assez abouti, passer ça sur Gladys

4.4 Rédaction

Rédaction de la thèse.

5 Difficultés rencontrées

- Obtention résultats arrondi randomisé
– Qu'est-ce qui est significatif ?
– Problématique intrinsèque à la démarche de recherche
– les cas d'utilisations mettant en valeur l'arrondi randomisé : j'aurais du en avoir plus
– l'article aurait pu être publié selon Bijan, et non selon David : or moi la partie DNN m'intéresse beaucoup (voire de m'orienter là dedans plus tard) donc je tenais à l'intégrer à l'article et donc à continuer à chercher
- Perte massive de données au début de l'été 2018
- Difficulté de rédaction de l'article
– Méthode scientifique autant que de rédaction complexe à mettre en oeuvre
- Manque de recul, désorientation quant à la méthode de travail
– Désormais résolu, efforts canalisés et bien aiguillés

Références

[1] Nasrine Damouche, Matthieu Martel, and Alexandre Chapoutot. Amélioration à la compilation de la précision de programmes numériques. *8th Journées nationales du Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel, GDRGPL*, pages 27–34, 2016.

[2] David Defour. Fp-anr : A representation format to handle floating-point cancellation at run-time. In *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*, pages 76–83. IEEE, 2018.

[3] James Demmel, Yozo Hida, William Kahan, Xiaoye S. Li, Sonil Mukherjee, and E. Jason Riedy. Error bounds from extra-precise iterative refinement. *ACM Trans. Math. Softw.*, 32 :325–351, 2006.

[4] Christophe Denis, Pablo De Oliveira Castro, and Eric Petit. Verificarlo : checking floating point accuracy through monte carlo arithmetic. *arXiv preprint arXiv :1509.01347*, 2015.

[5] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1) :5–48, 1991.

[6] Max Goldstein. Significance arithmetic on a digital computer. *Communications of the ACM*, 6(3) :111–117, 1963.

[7] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[8] Fabienne Jézéquel and Jean-Marie Chesneaux. Cadna : a library for estimating round-off error propagation. *Computer Physics Communications*, 178(12) :933–955, 2008.

[9] Douglass Stott Parker. *Monte Carlo Arithmetic : exploiting randomness in floating-point arithmetic*. University of California (Los Angeles). Computer Science Department, 1997.

[10] Michèle Pichat and Jean Vignes. *Ingénierie du contrôle de la précision des calculs sur ordinateur*. Editions Technip, 1993.

[11] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[12] Jean Vignes. Discrete stochastic arithmetic for validating results of numerical software. *Numerical Algorithms*, 37(1-4) :377–390, 2004.

[13] Dan Zuras, Mike Cowlishaw, Alex Aiken, Matthew Applegate, David Bailey, Steve Bass, Dileep Bhandarkar, Mahesh Bhat, David Bindel, Sylvie Boldo, et al. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, October 2008.