Formalisation en (Coq + Flocq) de la norme euclidienne en arithmétique flottante Double-Double

Vincent Lefèvre[b] Nicolas Louvet[d] Jean-Michel Muller[a]
Joris Picot[c] Laurence Rideau[b]

[a]CNRS, [b]INRIA, [c]ENS de Lyon, [d]Univ. Lyon 1

Formalisation, Coq , Flocq

- Vérifier des preuves mathématiques à l'aide d'un assistant de preuve: Coq

Formalisation, Coq , Flocq

- Vérifier des preuves mathématiques à l'aide d'un assistant de preuve: Coq
- Coq (Calculus of Constructions) : outil interactif de construction et de vérification de preuves

Formalisation, Coq , Flocq

- Vérifier des preuves mathématiques à l'aide d'un assistant de preuve: Coq
- Coq (Calculus of Constructions) : outil interactif de construction et de vérification de preuves
- Flocq : Bibliothèque en Coq dédiée à l'arithmétique flottante (G. Melquiond et S. Boldo)
  - différents modèles de représentation des nombres flottants
  - Définitions, Enoncés de proprietés et de théorèmes, Preuves

Formalisation, Coq , Flocq

- Vérifier des preuves mathématiques à l'aide d'un assistant de preuve: Coq
- Coq (Calculus of Constructions) : outil interactif de construction et de vérification de preuves
- Flocq : Bibliothèque en Coq dédiée à l'arithmétique flottante (G. Melquiond et S. Boldo)
  - différents modèles de représentation des nombres flottants
  - Définitions, Enoncés de proprietés et de théorèmes, Preuves
- But: Gain de confiance dans des preuves souvent très longues et calculatoires, en particulier pour des usages ultérieurs des algorithmes et des bornes d'erreur.

Briques de base

Algorithmes calculant des opérations binaires sur des arguments flottants

- 2Sum(a, b), Fast2Sum(a, b), Fast2Mult(a, b) traités en partie par Flocq
- Résultat exact sous la forme de 2 FP: arrondi + erreur

Briques de base

Algorithmes calculant des opérations binaires sur des arguments flottants

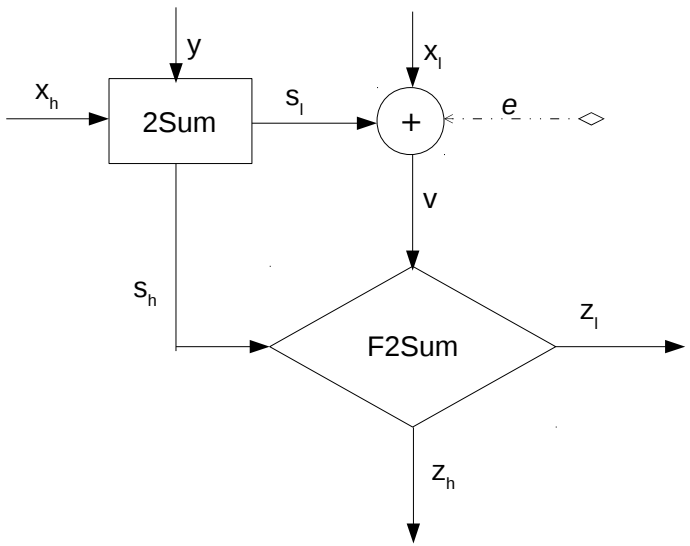
- 2Sum(a, b), Fast2Sum(a, b), Fast2Mult(a, b) traités en partie par Flocq
- Résultat exact sous la forme de 2 FP: arrondi + erreur
- Rappel: x est un nombre double-double (DW): représenté par la somme $x_h + x_\ell$ de deux FP x_h et x_ℓ tels que $x_h = \mathrm{RN}(x)$

Arithmétique pour les double-double (DW)

- Différents algorithmes de base (addition, multiplication et division d'un DW et d'un FP ou de 2 DW)
  proposés par M. Joldes, J.-M. Muller et V. Popescu en 2017
- Pour chaque algorithme:
  - une approximation (double-double) de l'opération sur les opérandes x et y, tels que $x = (x_h, x_\ell)$ est un DW et y est un FP ou un DW (y_h, y_ℓ).
  - une borne d'erreur
- Formalisés en Coq et amendés avec J.-M. Muller en 2020

Arithmetique pour les double-doubles: addition

DWPlusFP(x_h, x_ℓ, y): calcule une approximation de $(x_h, x_\ell) + y$, tels que $x = (x_h, x_\ell)$ est un DW et y est un FP.



- Borne d'erreur relative:

$$\frac{2 \cdot u^2}{1 - 2u} = 2u^2 + 4u^3 + 8u^4 + \cdots$$

- si x et y sont positifs, la borne devient u^2.

Résultats formalisés en Coq.

Racine carrée d'un DW

SQRTDWtoFP(x_h, x_ℓ): calcule une approximation de la racine carrée d'un DW (x_h, x_ℓ), retourne un FP z

- Preuve papier de la correction de l'algo et d'une borne d'erreur:
  - sans débordements
  - il ne peut y avoir d'overflow
  - un underflow est sans conséquence sur la correction de l'algorithme et de la borne d'erreur.

Racine carrée d'un DW

SQRTDWtoFP(x_h, x_ℓ): calcule une approximation de la racine carrée d'un DW (x_h, x_ℓ), retourne un FP z

- Preuve papier de la correction de l'algo et d'une borne d'erreur:
  - sans débordements
  - il ne peut y avoir d'overflow
  - un underflow est sans conséquence sur la correction de l'algorithme et de la borne d'erreur.
- Flocq propose différents modèles de représentation des nombres flottants, les définitions et les preuves associées
  - format FLX : sans restrictions sur les exposants, utilisé pour la preuve "sans débordements"
  - traitement des overflows: vérification que tous les nombres impliqués dans le calcul sont plus petits que Ω
  - traitement des underflows: format FLT avec une borne inférieure sur les exposants, propriétés et preuves sur les nombres flottants normaux et sous-normaux

Formalisation complète de toutes les preuves sur la racine carrée.

Calcul *séquentiel* d'une somme de carrés en arithmétique double-double

Algorithm 1 Sequential computation of $\sum_{i=0}^{n-1} a_i^2$ assuming no under/overflow.

1. For $i = 0 \ldots n-1$, compute $(y_i^h, y_i^\ell) = \mathrm{Fast2Mult}(a_i, a_i)$.
   (gives $a_i^2 = y_i^h + y_i^\ell$).
2. Accumulate the terms y_i^h in DW arithmetic: starting from $(x_1^h, x_1^\ell) = \mathrm{2Sum}(y_0^h, y_1^h)$, for $i = 2 \ldots n-1$, compute $(x_i^h, x_i^\ell) = \mathrm{DWPlusFP}(x_{i-1}^h, x_{i-1}^\ell, y_i^h)$.
3. Accumulate the terms y_i^ℓ in FP arithmetic: for $i = 0 \ldots n-2$, compute $\sigma_{i+1} = \mathrm{RN}(\sigma_i + y_{i+1}^\ell)$, with $\sigma_0 = y_0^\ell$.
4. Obtain the approximation to (S_h, S_ℓ) to $\sum_{i=0}^{n-1} a_i^2$ as

$$(S_h, S_\ell) = \mathrm{DWPlusFP}(x_{n-1}^h, x_{n-1}^\ell, \sigma_{n-1}).$$

Calcul par blocs d'une somme de carrés en double-double

- the a_i are separated into k blocks of m numbers, with $n = k \times m$;
- parallelizing the calculation & obtaining a more accurate result;
- block j ($j = 0, \ldots, k-1$) contains $a_{mj}, a_{mj+1}, \ldots, a_{m(j+1)-1}$.

Algorithm 2 Blockwise computation of $\sum_{i=0}^{n-1} a_i^2$ assuming no under/overflow.

---

1. for $j = 0, 1, \ldots, k-1$, compute an approximation (Z_j^h, Z_j^ℓ) to $\sum_{i=mj}^{m(j+1)-1} a_i^2$ using the sequential summation algorithm applied to $a_{mj}, a_{mj+1}, a_{mj+2}, \ldots, a_{m(j+1)-1}$;
2. accumulate the terms Z_j^h in DW arithmetic, i.e., starting from $(\Sigma_1^h, \Sigma_1^\ell) = 2\mathrm{Sum}(Z_0^h, Z_1^h)$, iteratively compute, for $j = 2 \ldots k-1$ the terms $(\Sigma_j^h, \Sigma_j^\ell) = \mathrm{DWPlusFP}\left(\Sigma_{j-1}^h, \Sigma_{j-1}^\ell, Z_j^h\right)$;
3. accumulate the terms Z_j^ℓ using the conventional "recursive" summation, i.e., for $j = 0 \ldots k-2$, compute $\tau_{j+1} = \mathrm{RN}(\tau_j + Z_{j+1}^\ell)$, with $\tau_0 = Z_0^\ell$;
4. obtain the approximation (S_h, S_ℓ) to $\sum_{i=0}^{n-1} a_i^2$ as

$$(S_h, S_\ell) = \mathrm{DWPlusFP}\left(\Sigma_{k-1}^h, \Sigma_{k-1}^\ell, \tau_{k-1}\right).$$

Lemma 1 (Lange and Rump)

Soit $\mathbb{F}$ *un sous-ensemble de* $\mathbb{R}$ *et soit* $\tilde{+}$ *une opération dans* $\mathbb{F}$ *telle que* $\forall a, b \in \mathbb{F}, |(a\tilde{+}b) - (a+b)| \leq \min\{|a|, |b|\}$. *Soit* x_1, x_2, $\ldots$, x_n *des éléments de* $\mathbb{F}$, *on définit ainsi les nombres* s_i *et* ϵ_i *:*

$$\begin{aligned} s_1 &= x_1, \\ s_i &= x_i \tilde{+} s_{i-1} = (x_i + s_{i-1})(1+\epsilon_i) \quad \text{pour } i = 2, \ldots, n. \end{aligned}$$

On a alors: $\left|s_n - \sum_{i=1}^{n} x_i\right| \leq \sum_{i=2}^{n} |\epsilon_i| \cdot \sum_{i=1}^{n} |x_i|$.

Lemma 1 (Lange and Rump)

Soit $\mathbb{F}$ un sous-ensemble de $\mathbb{R}$ et soit $\tilde{+}$ une opération dans $\mathbb{F}$ telle que $\forall a, b \in \mathbb{F}, |(a\tilde{+}b) - (a+b)| \leq \min\{|a|, |b|\}$. Soit x_1, x_2, ..., x_n des éléments de $\mathbb{F}$, on définit ainsi les nombres s_i et ϵ_i:

$$\begin{aligned} s_1 &= x_1, \\ s_i &= x_i \tilde{+} s_{i-1} = (x_i + s_{i-1})(1+\epsilon_i) \quad \text{pour } i = 2, \ldots, n. \end{aligned}$$

On a alors: $\left|s_n - \sum_{i=1}^{n} x_i\right| \leq \sum_{i=2}^{n} |\epsilon_i| \cdot \sum_{i=1}^{n} |x_i|$.

Exemples d'opérations $\tilde{+}$:

- $a\tilde{+}b = RN(a+b)$ avec a et b FP
- $a\tilde{+}b = (DWPlusFP\ a\ b)$ avec a DW et b FP
- $a\tilde{+}b = (SloppyDWPlusDW\ a\ b)$ avec a et b DW

NB: il faut que $\tilde{+}$ vérifie la condition nécessaire du lemme

Le lemme de Lange&Rump a été formalisé en Coq

Norme Euclidienne

- Norme euclidienne d'un vecteur $(a_0, a_1, a_2, \ldots, a_{n-1})$ de nombres flottants (FP).

$$N = \sqrt{\sum_{i=0}^{n-1} a_i^2}$$

- Approximation de chaque étape du calcul par un double-double (DW):
  - les a_i^2 (Fast2Mult)
  - la sommation (séquentielle ou par blocs)
  - la racine carrée (retourne soit un DW soit un FP)
- Pour chaque étape, les théorèmes de correction et de bornes d'erreur ont été vérifiés avec Coq
- NB: formalisation de la sommation dans le cas sans débordements uniquement: calcul de S_{med}

Computing Euclidean norms barring underflow/overflow

Theorem 2

Assume all $a_i \in \mathrm{MED}$, *the sequential or blockwise summation (with* k *blocks of* m *elements) is used to compute* (S_h, S_ℓ) *and Algorithm SQRTDWtoFP is used to approximate* $\sqrt{S_h + S_\ell}$ *by a FP number* R. *Let*

$$\lambda(t) = (2t-1) + (t-1)u + (2t-2)u^2 + (t-1)u^3,$$

and define

$$\nu = \begin{cases} \lambda(n) & \text{with the sequential summation} \\ \lambda(k) + \lambda(m) + \lambda(k)\lambda(m) & \text{with the blockwise summation} \end{cases}$$

If $\nu < \frac{1}{2u}$, *then:*

$$\left| R - \sqrt{\sum_{i=0}^{n-1} a_i^2} \right| \leq \left(\frac{1}{2} + u \cdot \left(\frac{7}{4} + \frac{\nu}{1 - \nu \cdot u^2} \right) \right) \mathrm{ulp}\left(\sqrt{\sum_{i=0}^{n-1} a_i^2} \right). \qquad (1)$$

Erreurs révélées par le processus de formalisation

- Hypothèse de contraction du Lemme de Lange&Rump pour SloppyDWPlusDW
- Borne d'erreur dans la dernière étape de sommation

Conclusion

- Preuves du papier formellement vérifiées jusqu'au calcul de la norme euclidienne sans débordements (cas "MED")
- Grande confiance dans les résultats pour des utilisations futures
- De nouveaux résultats formalisés:
  - borne d'erreur de DWPlusFP dans le cas positif
  - la racine carrée pour les double-double
  - Theorème de Lange & Rump
- La formalisation permet des expérimentations:
  Genéralisation des preuves: preuves valables sans choix de fonction pour l'arrondi au plus proche