INRIA

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE

From Rounding Error Estimation to Automatic Correction with Automatic Differentiation

Philippe LANGLOIS

No 3967

May 2000

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--3967--FR+ENG

INRIA
RHÔNE-ALPES

From Rounding Error Estimation to Automatic Correction with Automatic Differentiation

Philippe LANGLOIS *

Thème 2 — Génie logiciel
et calcul symbolique
Projet Arénaire

Rapport de recherche n ° 3967 — May 2000 — ?? pages

Abstract: Using automatic differentiation (AD) to estimate the propagation of rounding errors in numerical algorithms is classic. We propose a new application of AD to roundoff analysis providing an automatic correction of the first order effect of the elementary rounding errors. We present the main characteristics of this method and significant examples of its application to improve the accuracy of computed results and/or the stability of the algorithm.

Key-words: Automatic error analysis, rounding error, floating point arithmetic, automatic differentiation.

(Résumé : tsvp)

* Email: `Philippe.Langlois@ens-lyon.fr`. This manuscript is also available as a research report of the Laboratoire de l'Informatique du Parallélisme, Ecole Nornale Supérieure de Lyon, France, `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

De l'estimation des erreurs d'arrondi à leur correction automatique par différentiation automatique

Résumé : Il est classique d'utiliser la différentatiation automatique (AD) pour estimer la propagation des erreurs d'arrondi dans les agorithmes numériques. Nous proposons une nouvelle application de l'AD à l'analyse des erreurs d'arrondi qui permet une correction automatique des l'effet du premier ordre des erreurs d'arrondi élémentaires. Nous présentons les principales caractéristiques de cette méthode et des applications significatives à l'amélioration de la précision de résultats calculés et/ou à la stabilité d'algorithme.

Mots-clé : Analyse d'erreur automatique, erreur d'arrondi, arithmetique flottante, differentiation automatique.

1 Introduction.

Rounding error analysis and automatic differentiation share a long common history at least since 25 years [1]. Iri emphasizes this topic in his survey that begins the first volume of this series [4]. He describes rounding error estimates simply computable with automatic differentiation in the reverse mode. These classic approaches yield *an absolute bound* or *a probabilistic estimate* of the first-order approximate of the final rounding error with respect to the elementary rounding errors introduced by the computation of intermediate variables [4], [6].

We propose a new linear approach, the CENA method. The difference with previous methods is that elementary rounding errors are not bounded nor modeled with random variables but computed. Thus we compute *a correcting term* to improve the accuracy of finite precision result.

In this paper, we focus on absolute bound limitations, the main features of the CENA method, automatic bounding of rounding errors in the automatic differentiation process (in reverse mode) and significant applications of the correcting method.

2 Linearization Methods and Bound Limitations.

Let us consider $\widehat{x}_N = fl(f(X))$ the floating point computation of $x_N = f(X)$ where f is a real function of the floating point vector $X = (\widehat{x}_1, \ldots, \widehat{x}_n)$. This computation introduces intermediate variables $\widehat{x}_k$ and associated elementary rounding errors δ_k that satisfy

$$\widehat{x}_k = fl(\widehat{x}_i \circ \widehat{x}_j) = \widehat{x}_i \circ \widehat{x}_j + \delta_k = x_k + \delta_k, \tag{1}$$

where $n+1 \leq k \leq N$, $1 \leq i \leq j < N$, $\circ \in \{+, -, \times, /, \sqrt{}\}$, $\widehat{x}_j \neq 0$ when $\circ = /$, and $\widehat{x}_i = 0$, $\widehat{x}_j \geq 0$ when $\circ = \sqrt{}$.

So, $\widehat{x}_N = \widehat{f}(X, \delta)$, *i.e.*, a function of data X and elementary rounding errors $\delta = (\delta_{n+1}, \ldots, \delta_N)$. The global rounding error is approximated to the first-order with respect to the elementary rounding errors and

$$\widehat{x}_N - x_N = \sum_{k=n+1}^{N} \frac{\partial \widehat{f}}{\partial \delta_k}(X, \delta) \cdot \delta_k \; - E_L = \Delta_L - E_L, \tag{2}$$

where E_L is the *linearization error* associated with Δ_L.

A first obvious limitation of linearization methods comes from the potential predominance in relation (2) of E_L which is generally unknown.

Classical deterministic application of relation (2) introduces another limitation when δ_k is bounded with (a function of) the floating point precision $\mathbf{u}$. Unsuitable bounds for $|\Delta_L|$ may be computed as the compensation of elementary rounding errors and exactly computed results are not taken into account.

We respectively illustrate these two cases in IEEE-754 single precision computing expressions $E_1 = (2^{50} + 1) - 1$ and $E_2 = (2^{25} \times 2^{25}) - 2^{50}$ (natural evaluation order). In both cases, the actual global error is equal to zero but absolute bounding yields $|\Delta_L| \leq 2^{27}$. Here $|\delta_k| \leq |\widehat{x}_k|\,\mathbf{u}$, with $\mathbf{u} = 2^{-24}$. Hence, absolute bounding satisfies $|\widehat{E}_i - E_i| = (2^{50} + 2^{50})\mathbf{u}$ $(i = 1, 2)$. It is worth noting that automatic bounding may yield very pessimistic results after few floating point operations.

3 The CENA Method.

3.1 Principles, Hypothesis and Applications.

With the CENA method, we do not bound but *compute* Δ_L to yield a *corrected result* $\overline{x_N}$ more accurate than initial $\widehat{x}_N$. This corrected result suffers from a truncation error E_L and rounding errors E_C. We *validate* the accuracy of $\overline{x_N}$ yielding a dynamic bound of this residual error for *an identified class of algorithms* that satisfies $E_L = 0$. In such conditions, we apply this automatic correction to *final* or *selected intermediate* computed results. This latter case is interesting for iterative algorithms (actual accuracy depends on the iterate) or when the algorithm stability depends on the accuracy of some intermediate variables. Current softwares implementing the method use overloading operators in Ada or Fortran 90.

3.2 Linear Correction and Linear Algorithms.

We *compute* the correcting factor $\widehat{\Delta}_L = fl(\Delta_L)$ and define the *corrected result* $\overline{x_N} = fl\left(\widehat{x}_N - \widehat{\Delta}_L\right)$. The correcting factor satisfies $\widehat{\Delta}_L = \Delta_L + E_C$, where E_C represents the rounding error introduced by the computation of the correction. The smaller is the *residual error* $\overline{x_N} - x_N = -(E_L + E_C)$, the more efficient is the correcting method, *i.e.*, the more accurate is the corrected result $\overline{x_N}$.

We define *a linear algorithm* as an algorithm that only contains the operations $\{+, -, \times, /, \sqrt{\ }\}$ and such that
- every multiplication $fl\,(\widehat{x}_i \times \widehat{x}_j)$ satisfies $\widehat{x}_i = x_i$ or $\widehat{x}_j = x_j$, and
- every division $fl\,(\widehat{x}_i / \widehat{x}_j)$ satisfies $\widehat{x}_j = x_j$, and
- every square root $fl\left(\sqrt{\widehat{x}_i}\right)$ satisfies $\widehat{x}_i = x_i$.

Linear algorithm are interesting because $\widehat{x}_N$ computed with a linear algorithm satisfies $E_L = 0$ [7]. Examples of linear algorithms are classic summation algorithms, inner product computation, polynomial evaluation using Horner's method and the resolution of triangular linear system.

3.3 Computing the Correcting Term $\widehat{\Delta}_L$.

The correcting term $\widehat{\Delta}_L$ involves computing partial derivatives and elementary rounding errors defined by relation (2). Merging these two computations is the original feature of the CENA method. The computation of the partial derivatives $\partial \widehat{f}/\partial \delta_k$ is classic; reverse mode of automatic differentiation is applied as in [4] or [11] for example. Elementary rounding errors δ_k are well known in the computer arithmetic community. In the following relations, $\delta_k[\circ, i, j] = fl(x_k) - x_k$ with $x_k = x_i \circ x_j$, and right-side expressions are computed ones (we drop hats and fl-s).

The elementary rounding error associated with addition, subtraction and multiplication is a computable floating point number [2], [5]. We have

$\delta_k[+, i, j] = (x_k - x_i) - x_j$ when $|x_i| \geq |x_j|$, and

$\delta_k[\times, i, j] = x_k - (x_i^U \times x_j^U) - [(x_i^U \times x_j^L) + (x_i^L \times x_j^U)] - (x_i^L \times x_j^L)$.

In this last computation, each factor x is split into x^U and x^L, its upper and lower "half-length" parts that depend on floating point arithmetic characteristics. For division and square root, approximate elementary rounding error $\widehat{\delta}_k$ and corresponding approximation bounds for $|\widehat{\delta}_k - \delta_k|$ are also computable [9], [8]. We have

$\widehat{\delta}_k[/, i, j] = \{(x_k \times x_j) - x_i - \delta_k[\times, k, j]\}/x_k$, and

$\widehat{\delta}_k[\sqrt{\ }, i, j] = \{(x_k \times x_k) - x_i + \delta_k[\times, k, k]\}/(2 \times x_k)$.

3.4 Bounding the Computing Error E_C.

The error E_C associated with $\widehat{\Delta}_L$ computation contains three sources of error. Approximation errors exist when $\widehat{\delta}_k$ is computed, *i.e.*, for division and square root. Rounding errors corrupt the final inner product in relation (2) and the automatic differentiation process. We dynamically bound each of these errors using Wilkinson's running error analysis [12].

Let us consider here the automatic bounding of rounding errors introduced in the AD process, *i.e.*, $|fl(D_k) - D_k|$ where $D_k = \partial \widehat{x}_N/\partial \delta_k$ and $fl(D_k)$ is the corresponding value computed with AD in reverse mode. The following algorithm describes the simultaneous computation of partial derivatives $fl(D_k)$ (lines 3 and 4) and associated error bound ε_k (line 5 and output).

For intermediate and output variables x_k such that $x_k = fl(x_i \circ x_j)$, $\mathsf{Vertices\_To}(x_k)$ returns intermediate or input variables x_i and x_j; f_{ik} bounds the elementary rounding error in the corresponding computation of $\partial x_k/\partial x_i$ that may be different to zero when $\circ = /$ or $\sqrt{\ }$. These values are naturaly computed in the construction stage of the reverse mode.

4 Accuracy Improvement and Algorithm Stabilization.

In this last section, we propose two significant applications of the CENA method. Final correction and accuracy improvement is illustrated with inner product computation. Correcting sensitive intermediate variables, we stabilize Newton's iteration applied to the computation

Input: $\partial x_k/\partial x_i\ (k=n+1,N), Dx_N=1, others=0$
1: **for** $k=N$ **to** $n+1$ **do**
2: **for** $i \in$ Vertices_To(x_k) **do**
3: $C_{ki}=\partial x_k/\partial x_i$
4: $Dx_i=Dx_i+Dx_k\times C_{ki}$
5: $e_i=e_i+|Dx_k\times C_{ki}|+|Dx_i|+e_k\times|C_{ki}|+f_{ik}\times|Dx_k|$
6: **end for**
7: **end for**
Output: $\partial\,\widehat{x}_N/\partial\delta_k=Dx_k,\ \varepsilon_k=\mathbf{u}\times e_k\ (k=n+1,N)$

of polynomial multiple root. Computation and correction are IEEE-754 single precision results [3]. In both cases, we compare the efficiency of the correcting method to the classic use of higher precision (that corresponds here to IEEE-754 double precision).

4.1 Correcting the Final Result.

We compute the inner product X^TY with $X=[1,2,\cdots,2^n,-1,-2,\cdots,-(2^n)]^T$ and $Y=[1,1,\cdots,1]^T$. Chosen bound for the vector size n guarantees no overflow. Exact result satisfies $X^TY=0.0$ but (for $n>n_{\mathbf{u}}$, a threshold that depends on the precision) IEEE-754 single and double precision computations suffer from a forward error that increases as a power of 2 when n increases (see Figure 1).

Applying the correcting method to the single precision computation, we have $\overline{X^TY}=0.0$ (that does not appear on the log-scale Figure 1). Therefore the corrected inner product is exact for every considered n. It is worth noting that computed B_{E_C} that bounds the computing error E_C proves that single precision results are *false* without knowing the exact result since $|\overline{X^TY}-X^TY|=|X^TY|\leq B_{E_C}\ll|fl\left(X^TY\right)|$.

4.2 Correcting Sensitive Intermediate Variables.

It is well known that rounding errors provides instable Newton's iteration when it is applied to compute a polynomial root of multiplicity $m>1$ [10]. Correcting the final result of an iterative process is inefficient but correcting well-chosen intermediate variables may stabilize the iteration.

We consider polynomial $P(x)=(x-1)^6$. Starting with $x_0=2.0$, we compute Newton's iterate x_k until the absolute residual $|P(x_k)|<\sigma'$. In the following computations, we *a priori* choose $\sigma'=O(\sigma)=1.0\times10^{-37}$ where $\sigma\approx1.2\times10^{-38}$ is the smallest non-zero positive number in IEEE-754 single precision.

After a reasonable beginning of convergence, both single and double precision iterations become unstable until stopping fortunately with a zero absolute residual (breakdown is observed for some other choices of initial x_0).

When polynomials P and P' are evaluated using Horner's scheme, the corrected Newton's iteration $x_{k+1}=x_k-\overline{P(x_k)}/\overline{P'(x_k)}, (k=0,1,\dots)$ is stable and converges with the math-

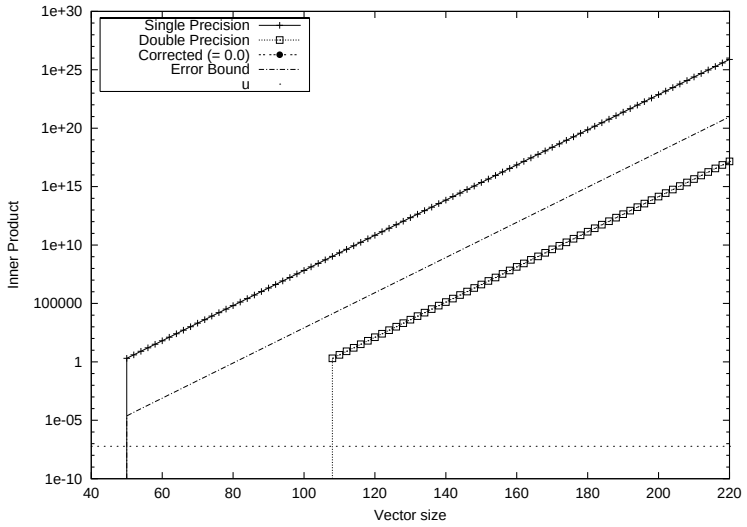

Figure 1: Computing $X^T Y = 0.0$.

ematically expected geometrical order $(m-1)/m$. The computed approximate x_k is more accurate than the classic estimation in $O(\mathbf{u}^m)$ where $\mathbf{u}$ is the precision of the computation. Figures 2 and 3 illustrate these properties.

References

[1] F. L. Bauer. Computational graphs and rounding errors. *SIAM J. Numer. Anal.*, 11(1):87–96, 1974.

[2] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math.*, 18:224–242, 1971.

[3] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.* Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

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

[5] Donald E. Knuth. *The Art of Computer Programming, Volume 2, Seminumerical Algorithms.* Addison-Wesley, Reading, MA, USA, third edition, 1998.

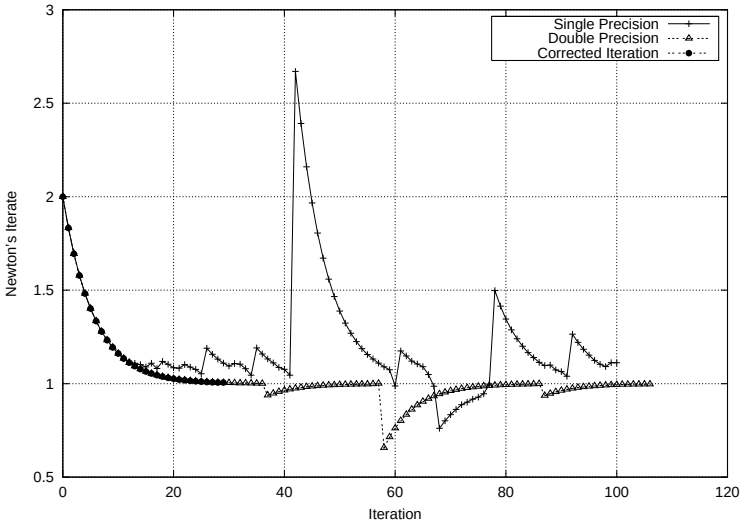

Figure 2: Newton's iterates until termination.

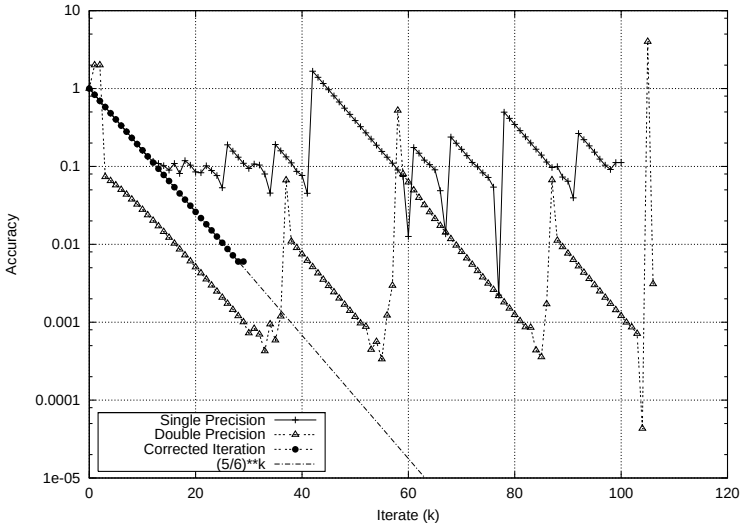

Figure 3: Accuracy of Newton's iterates until termination.

[6] Koichi Kubota. Padre2 - Fortran precompiler for automatic differentiation and estimates of rounding errors. In Martin Berz, Christian Bischof, George Corliss, and Andreas Griewank, editors, *Computational Differentiation. Techniques, Applications, and Tools*, pages 367–374. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[7] Philippe Langlois. Automatic linear correction of rounding errors. Numerical Analysis Report 355, Manchester Centre for Computational Mathematics, Manchester, England, November 1999. Also available as INRIA Research Report RR–3828, Dec. 1999.

[8] Philippe Langlois and Fabrice Nativel. Reduction and bounding of the rounding error in floating point arithmetic. *C.R. Acad. Sci. Paris, Série 1*, 327:781–786, 1998.

[9] Michèle Pichat. Correction d'une somme en arithmétique à virgule flottante. *Numer. Math.*, 19:400–406, 1972.

[10] Louis B. Rall. Convergence of the Newton process to multiple solutions. *Numer. Math.*, 9:25–37, 1966.

[11] Louis B. Rall and George F. Corliss. An introduction to automatic differentiation. In Martin Berz, Christian Bischof, George Corliss, and Andreas Griewank, editors, *Computational Differentiation. Techniques, Applications, and Tools*, pages 1–18. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[12] James H. Wilkinson. Error analysis revisited. *IMA Bulletin*, 22(11/12):192–200, 1986.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399