Analyse et implantation d'algorithmes rapides pour l'évaluation polynomiale sur les nombres flottants

Guillaume Revy

▶ **To cite this version:**

Guillaume Revy. Analyse et implantation d'algorithmes rapides pour l'évaluation polynomiale sur les nombres flottants. 2006. ensl-00119498

HAL Id: ensl-00119498

https://ens-lyon.hal.science/ensl-00119498

Preprint submitted on 10 Dec 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Analyse et implantation d'algorithmes rapides pour l'évaluation polynomiale sur les nombres flottants

Guillaume Revy 16 juin 2006

Rapport de DEA N° DEA2006-02

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

Analyse et implantation d'algorithmes rapides pour l'évaluation polynomiale sur les nombres flottants

Guillaume Revy

16 juin 2006

Résumé

L'évaluation de fonctions élémentaires reste un problème important en arithmétique des ordinateurs. Évaluer une telle fonction revient généralement à évaluer un polynôme qui l'approche au mieux. La méthode la plus utilisée, la méthode de Horner, permet d'évaluer un polynôme de degré n en n multiplications et n additions. Il existe par ailleurs d'autres méthodes, qui permettent d'évaluer des polynômes plus rapidement en nombre d'opérations que Horner. Cependant, ces méthodes nécessitent un préconditionnement préalable des polynômes à évaluer.

Pourquoi ne pas utiliser ces méthodes dans l'implantation de fonctions mathématiques ? Peut-on être plus rapide tout en restant aussi précis que Horner ?

Ce rapport montre que sous certaines conditions, ces méthodes peuvent fournir des erreurs comparables à celles obtenues par la méthode de Horner.

Mots-clés : arithmétique flottante, polynômes d'approximation, évaluation, préconditionnement, stabilité numérique, analyse d'erreurs et de complexité, fonctions élémentaires

Remerciements

Je tiens à remercier tout d'abord Claude-Pierre Jeannerod pour m'avoir fait confiance et donné l'opportunité d'effectuer mon stage de Master 2 au sein de l'équipe Arénaire du Laboratoire de l'Informatique et du Parallélisme (LIP) de l'École Normale Supérieure de Lyon (ENS-Lyon). Mais je tiens plus particulièrement à le remercier pour son immense disponibilité et son soutien tout au long de mon stage. Pendant ces cinq mois et demi, il a effectivement su m'aider et me guider dans mon travail.

Pour finir, je tiens à remercier Florent de Dinechin et Christoph Lauter pour leur aide, et plus généralement l'ensemble de l'équipe Arénaire pour leur accueil et leur soutien.

Table des matières

Introduction **1**

1 Rappels d'arithmétique virgule flottante **3**
1.1 Éléments de la norme IEEE-754 . . . 3
1.1.1 Nombres flottants normalisés IEEE-754 . . . 3
1.1.2 Quatre modes d'arrondis . . . 4
1.2 Le modèle flottant standard . . . 4
1.2.1 Notation préliminaire : unit roundoff . . . 4
1.2.2 Définitions . . . 4
1.2.3 Exemple d'analyse d'erreur d'arrondi dans le modèle flottant standard . . . 4
1.2.4 Erreurs d'arrondi . . . 5
1.3 Illustration par les méthodes de Horner et de Estrin . . . 5
1.3.1 Méthode de Horner . . . 5
1.3.2 Méthode de Estrin . . . 6
1.3.3 Observations numériques . . . 8

2 Algorithme de Knuth & Eve **9**
2.1 Description de l'algorithme . . . 9
2.1.1 Principe . . . 9
2.1.2 Premier cas : n impair . . . 9
2.1.3 Deuxième cas : n pair . . . 10
2.2 Le polynôme h n'a-t-il que des racines réelles ? . . . 11
2.2.1 Amélioration proposée par Eve . . . 11
2.2.2 Décalage du polynôme initial . . . 11
2.3 Préconditionnement et algorithme d'évaluation . . . 11
2.4 Analyse de complexité . . . 12
2.5 Analyse des erreurs d'arrondi . . . 12
2.6 Résultats numériques et observations . . . 13
2.6.1 Similitude avec Horner . . . 13
2.6.2 Effet du décalage . . . 13
2.6.3 Effet de la permutation . . . 16
2.6.4 Algorithmes hybrides . . . 17

3 Algorithme de Paterson & Stockmeyer **19**
3.1 Description . . . 19
3.2 Intérêt principal de cette approche . . . 19
3.3 Schéma d'évaluation et préconditionnement . . . 20
3.4 Analyse de complexité . . . 20
3.5 Analyse des erreurs d'arrondi . . . 21

3.6 Résultats numériques et observations 22
3.6.1 Similitude avec Estrin 22
3.6.2 Faut-il que les polynômes soient unitaires ? 22
3.6.3 Évaluation en le polynôme réciproque 23
3.6.4 Algorithmes hybrides 23

4 Implantation dans CR-Libm **25**
4.1 Fonctionnement de CR-Libm 25
4.1.1 Processus d'implantation d'une fonction élémentaire 25
4.1.2 Évaluation d'une fonction 25
4.2 Algorithme hybride pour le logarithme népérien 26
4.2.1 Comment est implanté le logarithme népérien ? 26
4.2.2 Solution proposée 26
4.3 Accélération apportée à la fonction $\arcsin(x)$ 27
4.3.1 Comment est implantée la fonction $\arcsin(x)$ dans CR-Libm ? 27
4.3.2 Présentation des algorithmes utilisés 27

Conclusion et perspectives **29**

Bibliographie **31**

Introduction

L'évaluation en logiciel et en virgule flottante de fonctions mathématiques reste aujourd'hui un domaine de recherche important en arithmétique des ordinateurs. De manière générale, l'évaluation de fonctions, notamment de fonctions élémentaires ($\ln(x)$, $\sin(x)$,...), sur un intervalle revient à évaluer un polynôme l'approchant au mieux sur cet intervalle, polynôme d'approximation. Mais peut-on alors évaluer des fonctions élémentaires de manière rapide et précise ?

En terme de précision, de nombreuses bibliothèques mathématiques fournissent un ensemble de fonctions élémentaires, mais elles ne garantissent pas toutes l'arrondi correct de leurs résultats, contrairement à CR-Libm [4], bibliothèque de fonctions mathématiques efficace et portable, dont les résultats sont prouvés correctement arrondis. Et en terme de rapidité, actuellement, la méthode la plus utilisée pour évaluer un polynôme en un point est la méthode de Horner. Introduite au XVII$^{\text{e}}$ siècle par Newton, elle permet l'évaluation d'un polynôme de degré n, $a(x) = \sum_{i=0}^{n} a_i x^i$, en n multiplications et n additions : $a(x) = \big(\dots(a_n x + a_{n-1})x + \cdots\big)x + a_0$. Cette méthode semble aujourd'hui être suffisamment précise.

Cependant, auparavant, une multiplication pouvait être beaucoup plus coûteuse en temps de calcul qu'une addition, ce qui a principalement motivé la mise en place de méthodes minimisant le nombre de multiplications. Il existe donc d'autres méthodes qui permettent l'évaluation de polynômes de manière plus rapide, notamment en un nombre de multiplications inférieur à celui nécessaire par la méthode de Horner (au détriment parfois du nombre d'additions). Ces évaluations plus rapides sont rendues possibles grâce au préconditionnement préalable du polynôme. Préconditionner un polynôme $a(x)$ donné par ses coefficients a_i consiste à calculer de nouveaux paramètres en fonction des a_i, puis à exprimer $a(x)$ en fonction de ces nouveaux paramètres. Cette première phase peut être coûteuse, mais sera effectuée une et une seule fois, le plus exactement possible, avant l'évaluation.

Pourquoi ne pas utiliser ces algorithmes rapides pour l'implantation de fonctions élémentaires dans les bibliothèques mathématiques, notamment dans CR-Libm ? L'implantation de ces méthodes efficaces fournit-elle des résultats en pratique aussi précis que la méthode de Horner et, sinon, perd-on vraiment en précision ?

Dans un premier chapitre, on rappellera les éléments d'arithmétique virgule flottante nécessaires à la compréhension du rapport, et on les illustrera avec la méthode de Horner. Ensuite, on comparera cette méthode à la méthode de Estrin. La méthode de Estrin permet l'évaluation d'un polynôme de degré n en $n + \log(n+1) - 1$ multiplications (et n additions). Elle nécessite donc un peu plus de multiplications que celle de Horner, mais l'évaluation peut être parallélisée, ce qui la rend un peu plus rapide que Horner en pratique.

Dans un deuxième chapitre, on présentera et on analysera une première méthode à base de préconditionnement, la méthode de Knuth & Eve. Cette méthode permet l'évaluation d'un polynôme de degré n en environ $\frac{n}{2}$ multiplications et n additions, et donc est toujours plus rapide que Horner. L'analyse d'erreur montre que, sous certaines conditions, cette méthode peut avoir la même borne d'erreur que l'algorithme de Horner. Cependant, le préconditionnement par cette méthode nécessite parfois un décalage du polynôme à évaluer, ce qui introduit généralement une perte de précision. En pratique, si aucun décalage n'est nécessaire, l'erreur maximale de l'algorithme de Knuth & Eve pourra être comparable à celle de Horner, voire meilleure. Autrement, pour un polynôme donné, plusieurs préconditionnements sont possibles, tous ne fournissant pas la même précision. La difficulté de cette méthode est alors de trouver un meilleur préconditionnement, celui minimisant l'erreur maximale sur un intervalle donné. On propose une méthode permettant de le déterminer.

Dans un troisième chapitre, on présentera la méthode de Paterson & Stockmeyer. Cette deuxième méthode à base de préconditionnement permet d'évaluer un polynôme de degré n en approximativement $\frac{n}{2} + \log(n)$ multiplications et $\frac{3n}{2}$ additions, suivant un schéma proche de celui de Estrin. L'analyse que l'on fait de cet algorithme fournit une borne d'erreur compararable à celle de Estrin. En pratique, on peut observer que cette méthode, sous certaines conditions, se comporte comme la méthode de Estrin.

Finalement, on présentera comment on a implanté ces algorithmes dans CR-Libm. La phase de préconditionnement a été implantée en Maple, puis, la phase d'évaluation, comme les codes nécessaires aux expérimentations, a été implantée en langage C. Dans ce quatrième chapitre, on expliquera tout d'abord le processus d'implantation d'une fonction élémentaire dans CR-Libm. Ensuite, on montrera comment on a pu, grâce à un algorithme hybride du type Paterson & Stockmeyer/Estrin, accélérer l'implantation de la fonction logarithme de CR-Libm, pourtant déjà relativement optimisée. Ce premier algorithme a ensuite été validé avec le générateur automatique de preuves Gappa [2]. On terminera par les accélérations apportées à la fonction $\arcsin(x)$ de CR-Libm. Pour accélérer cette fonction, on utilise notamment un schéma hybride de type Knuth & Eve/ Estrin. Certains de ces algorithmes ont également déjà été validés avec Gappa.

Chapitre 1

Rappels d'arithmétique virgule flottante

Dans la plupart des systèmes informatiques, les nombres réels sont approchés par des nombres à virgule flottante. La norme IEEE-754 [18, 17, 7] spécifie clairement le format de ces nombres en base 2, ainsi que le comportement des quatre opérations arithmétiques de base (addition, soustraction, multiplication et division) et de la racine carrée. Ce chapitre présente les notions d'arithmétique flottante nécessaires à la compréhension du rapport, et rappelle l'analyse en termes de complexité et d'erreurs d'arrondi des schémas d'évaluation usuels, Horner et Estrin.

1.1 Éléments de la norme IEEE-754

1.1.1 Nombres flottants normalisés IEEE-754

En base 2, un **nombre flottant** x est caractérisé par un **signe** s (0 ou 1), une **mantisse** m et un **exposant** e, de telle sorte que :

$$x = (-1)^s \times m \times 2^e. \tag{1.1}$$

La mantisse, sur p bits, est de la forme $m = x_0.x_1x_2\cdots x_{p-1}$, où les $x_i \in \{0,1\}$: le nombre est alors dit de **précision** p. L'exposant e, sur r bits, appartient à l'intervalle $[e_{min},e_{max}]$, avec $e_{min} = 2 - 2^{r-1}$ et $e_{max} = 2^{r-1} - 1$.

Il existe plusieurs formats de représentation des nombres flottants, de précisions différentes. La norme IEEE-754 en impose deux : les formats simple précision et double précision. La table 1.1 présente ces principaux formats.

Format	Simple précision	Double précision	Double précision étendue
Mantisse m	1 + 23 bits	1 + 52 bits	1 + 63 bits
Exposant e	8 bits	11 bits	15 bits
Précision p	24 bits	53 bits	64 bits
e_{min}	-127	-1022	-16382
e_{max}	128	1023	16382
u	$2^{-24} \approx 5.9 \times 10^{-8}$	$2^{-53} \approx 1.1 \times 10^{-16}$	$2^{-64} \approx 5.4 \times 10^{-20}$

TAB. 1.1 – Formats de représentation des nombres normalisés IEEE-754

1.1.2 Quatre modes d'arrondis

La norme IEEE-754 spécifie quatres modes d'arrondis : *vers le bas*, *vers le haut*, *vers zéro* et *au plus près*. Dans ce dernier mode d'arrondi, mode d'arrondi par défaut, un nombre réel situé entre deux nombres flottants sera arrondi vers le nombre flottant le plus proche, et s'il est à égale distance de ces deux nombres, on choisit celui dont la mantisse est paire.

1.2 Le modèle flottant standard

1.2.1 Notation préliminaire : unit roundoff

On utilisera dans ce rapport la quantité u, **unit roundoff**, définie de la manière suivante :

$$u = 2^{-p} = \frac{1}{2}ulp(1).$$

Cette quantité est la plus utilisée pour l'analyse d'erreur sur les nombres flottants [10]. Elle représente en fait la moitié de la distance entre le nombre flottant 1 et son successeur. En simple précision, $u = 2^{-24} \approx 5.9 \times 10^{-8}$ (Tab. 1.1).

1.2.2 Définitions

On note l'ensemble $\mathbb{F}$, ensemble des nombres de mantisse de n bits, de la manière suivante :

$$\mathbb{F} = \{(-1)^s \times m \times 2^e, \text{ avec } 2^{n-1} \leq |m| \leq 2^n - 1, \quad m, e \in \mathbb{Z}\} \cup \{0\}$$

Cet ensemble ne représente pas l'ensemble des nombres flottants. Il peut être vu comme un système *idéal* à travers lequel on se débarrasse de tout ce qui est contraignant dans le modèle flottant réel (bornes sur l'exposant, dépassements de capacité, nombres sous-normaux, ...). Ce modèle est en particulier vérifé par la norme IEEE-754.

Sur cet ensemble $\mathbb{F}$, on définit le **modèle flottant standard**. Ce modèle spécifie la précision des quatre opérations arithmétiques de base. Pour x,$y \in \mathbb{F}$ et $\mathtt{op} \in \{+, -, \times, \div\}$, on a :

$$fl(x \,\mathtt{op}\, y) = (x \,\mathtt{op}\, y)(1 + \delta) \quad \text{avec } \delta \in \mathbb{R} \text{ tel que } |\delta| \leq u \quad (u : \text{unit roundoff}). \tag{1.2}$$

On note $fl(\cdot)$, où $\cdot$ représente une expression, la valeur calculée de cette expression. D'après ce modèle, la valeur calculée $x \,\mathtt{op}\, y$ est aussi bonne que la valeur exacte arrondie. Cependant, lorque $x \,\mathtt{op}\, y$ est exacte ($x \,\mathtt{op}\, y \in \mathbb{F}$), le modèle ne spécifie pas que δ doit être nulle.

1.2.3 Exemple d'analyse d'erreur d'arrondi dans le modèle flottant standard

Dans le modèle flottant standard, une succession de calculs peut entraîner un ensemble d'erreurs, qui peuvent soit se compenser soit, le plus souvent, s'accumuler.

$$\begin{aligned} fl\big(x \times y + z\big) &= fl\big((x \times y)(1 + \delta_1) + z\big) \\ &= \big((x \times y)(1 + \delta_1) + z\big) \times (1 + \delta_2) \\ &= (x \times y)(1 + \delta_1)(1 + \delta_2) + z(1 + \delta_2), \quad \text{avec} \quad |\delta_1|, |\delta_2| \leq u \quad \text{d'après (1.2).} \end{aligned}$$

Pour simplifier l'écriture des bornes d'erreurs, on utilisera les hypothèses et définitions classiques suivantes [10]. Si $|\delta_i| \leq u$ et $nu < 1$, on a :

$$|\theta_n| \leq \gamma_n \quad \text{avec} \quad 1 + \theta_n := \prod_{i=1}^{n}(1 + \delta_i) \quad \text{et} \quad \gamma_n := \frac{nu}{1 - nu} \tag{1.3}$$

Par exemple, en double précision, on aura : $\gamma_6 = \frac{6u}{1-6u} \approx 3 \times 2^{-52} \approx 6.7 \times 10^{-16}$. On note aussi <n> pour $1 + \theta_n$ [10]. Ainsi, (1.3) s'écrit également :

$$\begin{aligned}(x \times y)(1+\delta_1)(1+\delta_2) + z(1+\delta_2) &= (x \times y)(1+\theta_2) + z(1+\theta_1) \\ &= (x \times y)\langle 2\rangle + z\langle 1\rangle.\end{aligned}$$

1.2.4 Erreurs d'arrondi

L'analyse des erreurs d'arrondi a pour objectif de majorer l'erreur effectuée lors de l'évaluation, dans le modèle flottant standard, d'un polynôme par une méthode d'évaluation (Horner, Estrin...), par rapport à la valeur réelle. On peut distinguer deux types d'erreur : l'erreur absolue et l'erreur relative. On note $\widehat{r}$ la valeur approchée d'un polynôme a en un point x, calculée par une méthode d'évaluation.

L'**erreur absolue** est $|\widehat{r} - a|$, et si $a(x) \neq 0$, l'**erreur relative** est $\frac{|\widehat{r}-a(x)|}{|a(x)|}$.

1.3 Illustration par les méthodes de Horner et de Estrin

La méthode de Horner est l'algorithme le plus classique pour évaluer un polynôme en un point. Son comportement numérique, sur les nombres flottants en tenant compte des erreurs d'arrondi, est relativement bien compris, et il s'avère qu'en termes de précision, ce schéma semble satisfaisant.

1.3.1 Méthode de Horner

Schéma d'évaluation. La méthode de Horner permet l'évaluation d'un polynôme de degré n en n multiplications et n additions. On considère un polynôme a de degré n. Son schéma d'évaluation par la méthode de Horner est illustrée par (1.4).

$$a(x) = \big(\dots(a_n x + a_{n-1})x + \cdots\big)x + a_0. \tag{1.4}$$

L'algorithme d'évaluation est présenté par l'algorithme 1 ci-dessous.

Algorithme 1 Horner

Entrées : $a_0, \cdots, a_n, x$
Sorties : $a(x) = \sum_{i=0}^{n} a_i x^i$

```
r ← a_n ;
pour i de n − 1 à 0 faire
    r ← r × x + a_i ;
fin pour
retourner r ;
```

Analyse des erreurs d'arrondi. Higham [10] présente une méthode d'analyse d'erreur dans le modèle flottant standard permettant de majorer les erreurs absolue et relative dues à l'utilisation d'une méthode d'évaluation. On l'illustre ici avec un polynôme a de degré 3. Son évaluation en x par la méthode de Horner s'écrit :

1. $r_1 \leftarrow a_3$
2. $r_2 \leftarrow r_1 x + a_2$
3. $r_3 \leftarrow r_2 x + a_1$
4. $r_4 \leftarrow r_3 x + a_0$

Dans le modèle flottant standard, le schéma précédent devient :

1.	$\widehat{r}_1 = a_3$
2.	$\widehat{r}_2 = a_5 x <2> + a_4 <1>$
3.	$\widehat{r}_3 = a_5 x^2 <4> + a_4 x <3> + a_3 <1>$
4.	$\widehat{r}_4 = a_5 x^3 <6> + a_4 x^2 <5> + a_3 x <3> + a_2 <1>$

Ici $\widehat{r}_4$ est la valeur effectivement calculée pour $a(x)$. On a alors :

$$\begin{aligned} \widehat{r}_4 &= a_3 x^3 <6> + a_2 x^2 <5> + a_1 x <3> + a_0 <1> \\ &= a_3 x^3 (1+\theta_6) + a_2 x^2 (1+\theta_5) + a_1 x (1+\theta_3) + a_0 (1+\theta_1). \end{aligned} \tag{1.5}$$

Or, d'après (1.3), on a $|\theta_i| \leq \gamma_i$. De (1.5), on peut donc dans un premier temps majorer l'erreur absolue :

$$|\widehat{r}_4 - a(x)| \leq \gamma_6 |a_3| \cdot |x^3| + \gamma_5 |a_2| \cdot |x^2| + \gamma_3 |a_1| \cdot |x| + \gamma_1 |a_0|$$

Higham montre que $\gamma_j \geq \gamma_i$ pour $j \geq i$ [10], donc $\gamma_6 \geq \gamma_5 \geq \gamma_3 \geq \gamma_1$. On peut conclure :

$$|\widehat{r}_4 - a(x)| \leq \gamma_6 \tilde{a}(|x|) \quad \text{avec} \quad \tilde{a}(x) = \sum_{i=0}^{3} |a_i| x^i.$$

De manière générale, pour un polynôme de degré n, on aura :

$$|\widehat{r} - a(x)| \leq \gamma_{2n} \tilde{a}(|x|). \tag{1.6}$$

Si $a(x) \neq 0$, on pourra de plus majorer l'erreur relative de la manière suivante :

$$\frac{|\widehat{r} - a(x)|}{|a(x)|} \leq \gamma_{2n} \frac{\tilde{a}(|x|)}{|a(x)|}. \tag{1.7}$$

tout en remarquant que, si l'évaluation se fait en une valeur de x proche d'une racine de a, cette borne peut être très grande.

On appelle $\frac{\tilde{a}(|x|)}{|a(x)|}$ le conditionnement de l'évaluation d'un polynôme a en x lorsque a est donné par ses coefficients a_i ($\frac{\tilde{a}(|x|)}{|a(x)|} \geq 1$) [10]. Cependant si $a_i \geq 0$ pour tout i et si $x \geq 0$, ou si $(-1)^i a_i \geq 0$ pour tout i et si $x \leq 0$, le conditionnement sera égal à 1.

Le problème de l'évaluation d'un polynôme a en x sera dit « bien conditionné » si la quantité $\frac{\tilde{a}(|x|)}{|a(x)|}$ est « petite ».

1.3.2 Méthode de Estrin

Description. La méthode de Horner est une méthode précise, dont le résultat à l'itération i dépend directement de celui à l'itération $i-1$. D'autres méthodes ont été mises en place, exploitant la stratégie « diviser pour régner ». C'est le cas de la méthode de Estrin qui permet d'évaluer un polynôme de degré n en $n + \log(n+1) - 1$ multiplications et n additions [13]. Pour simplifier, on considère à présent pour cette méthode des polynômes de degré $n = 2^p - 1$. Un polynôme dont le degré n'est pas de la forme $2^p - 1$ pourra être découpé en polynômes de degré $2^i - 1$, selon l'écriture binaire de n, évalués séparément par cette méthode.

Schéma d'évaluation et complexité. On considère un polynôme a de degré 7. Son évaluation par la méthode de Estrin est la suivante :

$$a(x) = \big((a_7x + a_6) \cdot x^2 + (a_5x + a_4)\big) \cdot x^4 + \big((a_3x + a_2) \cdot x^2 + (a_1x + a_0)\big). \tag{1.8}$$

En calculant les puissances successives de x, l'évaluation nécessite 9 multiplications et 7 additions. Son schéma d'évaluation peut alors être représenté sous la forme d'un arbre binaire (Fig. 1.1).

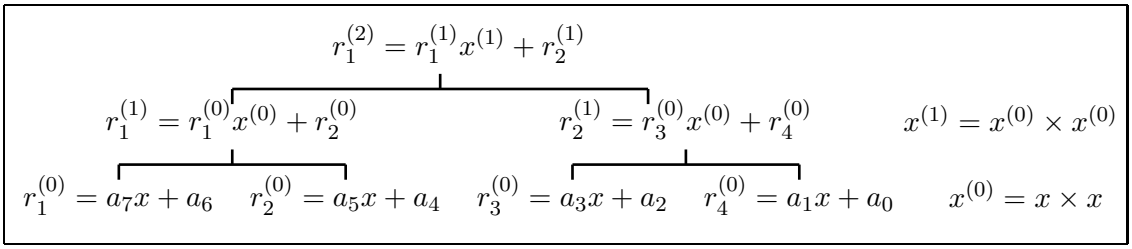


FIG. 1.1 – Évaluation par la méthode de Estrin

En notant $C(p)$ le coût d'évaluation d'un polynôme de degré $n = 2^p - 1$, sachant que l'on connaît les puissances de x $(x^2, x^4, \ldots x^{2^{p-1}})$, on obtient :

$$\begin{cases} C(1) &= 1 \text{ multiplication } + 1 \text{ addition} \\ C(p) &= 2C(p-1) + 1 \text{ multiplication } + 1 \text{ addition} \end{cases}$$

De plus, le calcul des puissances successives de x nécessite $\log(n+1) - 1$ multiplications (soit $p - 1$ multiplications). D'où un total de $n + p - 1$ multiplications et n additions. Certes, $p - 1$ multiplications supplémentaires sont nécessaires par rapport à la méthode de Horner, mais l'évaluation parallélisée rend ce schéma plus rapide en pratique (Fig. 1.1).

Analyse des erreurs d'arrondi. On considère pour commencer un exemple en degré 3 : $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Dans le modèle flottant standard, l'évaluation de ce polynôme par la méthode de Estrin est la suivante :

1.	$\widehat{r}_1 = a_3x<2> + a_2<1>$
2.	$\widehat{r}_2 = a_1x<2> + a_0<1>$
3.	$\widehat{x}_2 = x^2<1>$
4.	$\widehat{r}_3 = \widehat{r}_1\widehat{x}_2<2> + \widehat{r}_2<1> = (a_3x < 5 > + a_2 < 4 >)x^2 + (a_1<2>x + a_0<1>)$

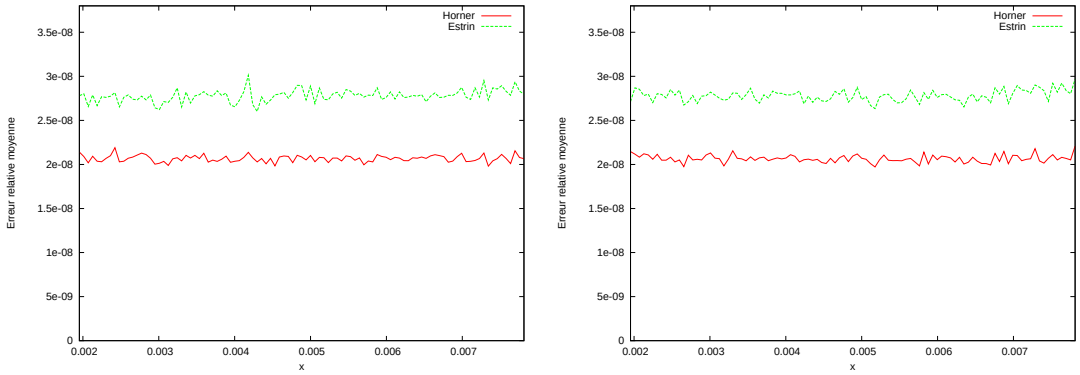
L'erreur absolue est donc définie de la manière suivante : $|\widehat{r}_4 - a(x)| \leq \gamma_5 \sum_{i=0}^{3} |a_i||x^i|$. De manière générale, les erreurs relative et absolue sont :

$$|\widehat{r} - a(x)| \leq \gamma_{n+\log(n+1)}\tilde{a}(|x|) \quad \text{et} \quad \frac{|\widehat{r} - a(x)|}{|a(x)|} \leq \gamma_{n+\log(n+1)}\frac{\tilde{a}(|x|)}{|a(x)|}.$$

On peut observer que si $a_i \geq 0$, pour tout i, et si $x \geq 0$, ou si $(-1)^i a_i \geq 0$ et si $x \leq 0$, alors $\tilde{a}(|x|)/|a(x)| = 1$. Ces conditions sont les mêmes que pour la méthode de Horner. Les preuves des bornes d'erreurs d'arrondi sont présentées en [11].

1.3.3 Observations numériques

En pratique, on peut observer que la méthode de Horner se comporte mieux que ce que prédit la théorie [10]. Il s'avère que l'on peut également observer cela pour la méthode de Estrin. La figure 1.2 montre l'erreur relative moyenne pour l'évaluation de 1000 polynômes, tous bien conditionnés pour Horner et Estrin, de degré 15 et 31 évalués par les méthodes de Horner et Estrin. Les polynômes ont été générés aléatoirement avec Maple. Les calculs ont été effectués en simple précision, et les erreurs relatives déterminées par rapport à la valeur en double précision avec Horner considérée comme exacte.



(a) polynôme de degré 15 (b) polynôme de degré 31

FIG. 1.2 – Erreur relative moyenne par Estrin

En revanche, même si la borne d'erreur relative est meilleure pour Estrin que pour Horner, il s'avère qu'en pratique Estrin est un peu moins précis que Horner (mais reste du même ordre de grandeur).

Pour finir, on peut observer que dans certains cas, l'erreur relative maximale par la méthode de Estrin est inférieure à celle par Horner. La figure 1.3 montre l'erreur relative de l'évaluation d'un polynôme de degré 15 par les méthodes de Horner et Estrin.

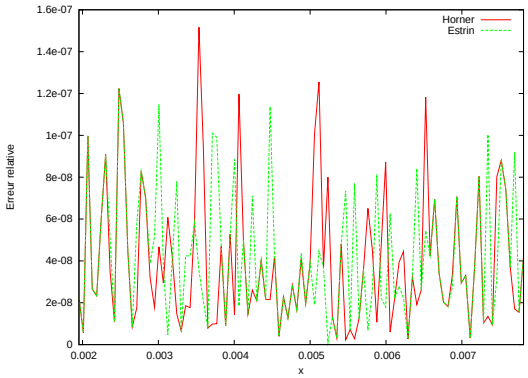


FIG. 1.3 – Erreurs relatives par Horner et Estrin

Chapitre 2

Algorithme de Knuth & Eve

Dans les années 1960, Knuth a proposé un nouveau schéma d'évaluation de polynômes [12], idée complétée par Eve quelques années plus tard [5]. Cette méthode se base sur le découpage en partie paire et partie impaire du polynôme à évaluer, et utilise un préconditionnement non rationnel.

2.1 Description de l'algorithme

2.1.1 Principe

On considère un polynôme a de degré n : $a(x) = \sum_{i=0}^{n} a_i x^i$. L'idée principale consiste à découper le polynôme en une partie paire g et une partie impaire h. En notant $y = x^2$, on obtient :

$$a(x) = g(y) + x \cdot h(y)$$

avec $g(y) = \sum_{i=0}^{\lfloor n/2 \rfloor} a_{2i} y^i$ et $h(y) = \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} a_{2i+1} y^i$. On note $l = \deg(g)$ et $m = \deg(h)$.

Si g (respectivement h) est un polynôme nul ou une constante non nulle, l'évaluation de $a(x)$ revient à évaluer h (respectivement g) en y, avec une multiplication supplémentaire, et une addition dans le cas où g est une constante non nulle (respectivement avec une multiplication et une addition supplémentaire si h est une constante non nulle). Dans ces deux cas, on peut remarquer que l'on divise approximativement le nombre de multiplications et d'additions par 2 par rapport à la méthode de Horner.

Sinon, on suppose tout d'abord que h n'a que des racines réelles, notées $\alpha_1, \cdots, \alpha_m$:

$$h(x) = a_{2m+1}(x - \alpha_m)(x - \alpha_{m-1}) \cdots (x - \alpha_1).$$

On étudiera en section 2.2 comment traiter le cas où h a des racines complexes.

2.1.2 Premier cas : n impair

On suppose ici que $n = 2m+1$. Donc, dans ce cas, $l \leq m = \frac{n-1}{2}$. En divisant g par h, on obtient un quotient q et un reste r de telle sorte que :

$$g(y) = h(y) \cdot q(y) + r(y), \quad \text{avec} \quad \deg(q) = l - m \quad \text{et} \quad \deg(r) =: d < m.$$

Dans la suite, on se limitera au cas générique où $l = m = d + 1$ (le cas général se traite de la même façon). On a alors $q(y)$ constant, égal à $\frac{a_{n-1}}{a_n}$.

On note $\beta_0, \beta_1 \dots \beta_{m-1}$ les coefficients de la forme de Newton de r par rapport aux α_i. On a alors :

$$r(y) = \sum_{i=0}^{m-1} \beta_i \prod_{j=1}^{i} (y - \alpha_j).$$

Les β_i, pour $0 \leq i < m$, peuvent être obtenus par divisions euclidiennes successives de g par $y - \alpha_{i+1}$.

En conclusion, on obtient :

$$\begin{aligned} a(x) &= g(y) + x \cdot h(y) = h(y) \cdot q(y) + r(y) + x \cdot h(y) = (q(y) + x) \cdot h(y) + r(y) \\ &= a_{2m+1}\big(q(y) + x\big) \cdot \prod_{i=1}^{m} (y - \alpha_i) + \sum_{i=0}^{m-1} \beta_i \prod_{j=1}^{i} (y - \alpha_j) \\ &= \big((a_{2m+1}(q(y) + x)(y - \alpha_m) + \beta_{m-1}) \cdots\big)(y - \alpha_1) + \beta_0. \end{aligned}$$

Par exemple, on considère un polynôme a de degré 5 : $a(x) = \sum_{i=0}^{5} a_i x^i$. Les polynômes g et h sont de la forme :

$$g(y) = a_4 y^2 + a_2 y + a_0 \quad \text{et} \quad h(y) = a_5 y^2 + a_3 y + a_1 = a_5 (y - \alpha_2)(y - \alpha_1).$$

avec $l = m = 2$. Par division de g par h, on obtient $g(y) = q(y) \cdot h(y) + r(y)$, avec :

$$q(y) = \frac{a_4}{a_5} \quad \text{et} \quad r(y) = \beta_1 (y - \alpha_1) + \beta_0.$$

Finalement, le schéma d'évaluation de a par l'algorithme de Knuth & Eve est le suivant :

$$a(x) = \big((a_5 x + a_4)(y - \alpha_2) + \beta_1\big)(y - \alpha_1) + \beta_0. \tag{2.1}$$

L'évaluation nécessite alors 4 multiplications et 5 additions, contre 5 multiplications et 5 additions avec la méthode de Horner.

2.1.3 Deuxième cas : n pair

Pour le cas n pair, la théorie présentée ci-dessus reste valable. Mais on a dans ce cas $m < l$, et plus particulièrement $l = m + 1 = \frac{n}{2}$.

Ensuite, par divisions successives de g par $y - \alpha_i$, on peut déterminer les coefficients β_i, pour $1 \leq i \leq d + 1$.

Par exemple, on considère le polynôme a de degré 6 : $a(x) = \sum_{i=0}^{6} a_i x^i$. Les polynômes g et h sont de la forme :

$$g(y) = a_6 y^3 + a_4 y^2 + a_4 y + a_0 \quad \text{et} \quad h(y) = a_5 y^2 + a_3 y + a_1 = a_5 (y - \alpha_2)(y - \alpha_1).$$

avec $l = 3$ et $m = 2$. Par division de g par h, on obtient $g(y) = q(y) \cdot h(y) + r(y)$, avec :

$$q(y) = q_1 y + q_0 \quad \text{et} \quad r(y) = \beta_1 \cdot (y - \alpha_1) + \beta_0.$$

Finalement, le schéma d'évaluation de a par l'algorithme de Knuth & Eve est :

$$a(x) \quad = \quad \Big(a_5\big((q_1 y + q_0) + x\big)(y - \alpha_2) + \beta_1\Big)(y - \alpha_1) + \beta_0.$$

Dans ce cas, si $l = m+1$, le quotient sera un polynôme de degré 1 en y. L'évaluation nécessitera 5 multiplications et 6 additions, contre 6 multiplications et 6 additions par la méthode de Horner.

2.2 Le polynôme h n'a-t-il que des racines réelles ?

2.2.1 Amélioration proposée par Eve

On a considéré jusqu'à présent que le polynôme h, partie impaire du polynôme à évaluer, n'avait que des racines réelles. C'est d'ailleurs une condition nécessaire pour que le polynôme puisse être évalué par cet algorithme. Mais qu'est-ce qui assure cette condition ?

En 1964, Eve [5] a montré que si le polynôme a de degré n, à évaluer, a au moins $n-1$ racines dont les parties réelles sont toutes non négatives ou toutes non positives, alors h n'a que des racines réelles.

Cependant, si ce n'est pas le cas, il est possible de ramener le polynôme a à un polynôme f respectant cette condition par décalage, puis d'évaluer f par l'algorithme de Knuth & Eve. En notant c le décalage, on obtient : $f(x) = a(x - c)$. L'évaluation se fera non plus en x mais en $x + c$, car $a(x) = f(x + c)$.

2.2.2 Décalage du polynôme initial

Méthode proposée par Eve. La première méthode de décalage consiste à déterminer de façon approchée les n racines de a (par exemple via l'outil Maple). On note r_l la partie réelle de la deuxième racine la plus petite et r_r la partie réelle de la deuxième racine la plus grande. Si a n'a pas au moins $n-1$ racines toutes non négatives, le décalage c est alors égal $-r_l$ (on le notera *first* par la suite), ou bien si a n'a pas au moins $n-1$ racines toutes non positives le décalage est alors égal $-r_r$ (on le notera *last*). On peut remarquer que les deux décalages sont possibles, il suffira seulement de choisir celui garantissant une meilleure stabilité numérique.

Méthode proposée par Knuth. Cette méthode permet de déterminer un décalage c et une valeur de α_1 pour lesquels β_0 sera nul [13], ce qui diminue le nombre d'additions de 1. Si $r_1 = a_1 + b_1 i$ est la racine ayant la plus petite (ou bien la plus grande) partie entière et $b_1 \neq 0$ alors $c = -a_1$ et $\alpha_1 = -b_1^2$. Sinon ($b_1 = 0$), si $r_2 = a_2 + b_2 i$ est la racine ayant la deuxième plus petite (ou bien la deuxième plus grande) partie entière et $b_2 \neq 0$ alors $c = -a_2$ et $\alpha_1 = -b_2^2$. Sinon, r_1 est la racine conjuguée de r_2 et dans ce cas, $c = -a$ et $\alpha_1 = b^2$.

Ces deux méthodes permettent de déterminer deux décalages. Mais on peut choisir un décalage plus grand si l'on veut que h n'ait que des racines à parties réelles toutes non négatives, ou un décalage plus petit si l'on veut que h n'ait que des racines à parties réelles toutes non positives.

2.3 Préconditionnement et algorithme d'évaluation

Le préconditionnement d'un polynôme a pour l'évaluation par la méthode de Knuth & Eve se décompose alors en 5 étapes :

1. calcul des racines de a
2. calcul du décalage c et calcul des coefficients f_i du polynôme décalé : $f(x) = \sum_{i=0}^{n} f_i x^i = a(x - c)$
3. calcul des coefficients des moitiés paire g et impaire h de f
4. calcul des racines α_i, toutes réelles, de h
5. détermination des β_i par divisions successives de g par $y - \alpha_{i+1}$

Cette phase de préconditionnement consiste alors à déterminer un ensemble de α_i et de β_i. Les α_i sont les racines de h, donc elles ne sont pas exactes, mais juste déterminées « le plus exactement possible ». Ensuite, les β_i sont calculés exactement en fonction des valeurs approchées des α_i. Une fois ces éléments obtenus, ils sont arrondis au plus près, pour pouvoir être représentables en machine au format considéré.

L'algorithme d'évaluation, pour n impair et a générique ($l = m = d + 1$), est alors présenté par l'algorithme 2.

Algorithme 2 Knuth & Eve

Entrées : c, f_n,f_{n-1},$\alpha_1, \cdots, \alpha_m$ et $\beta_0, \cdots, \beta_{m-1}$
Sorties : $a(x)$

```
s ← x + c ;
y ← s^2 ;
β_m ← f_n s + f_{n-1} ;
v_m ← β_m ;
pour i de m − 1 à 0 faire
    y_{i+1} ← y − α_{i+1} ;
    v_i ← v_{i+1} y_{i+1} + β_i ;
fin pour
retourner v_0 ;
```

2.4 Analyse de complexité

On considère un polynôme de degré $n \geq 3$ (si $n < 3$, on considère l'algorithme de Horner). En notant $C(d)$ le coût de l'évaluation d'un polynôme de degré d, on obtient :

$$\begin{cases} C(1) & = & 1 \text{ multiplication } /1 \text{ addition} \\ C(2) & = & 2 \text{ multiplications } /2 \text{ additions} \\ C(n),\ n \geq 3 & = & \frac{n+3}{2} \text{ multiplications } /n \text{ additions} \quad \text{si } n \text{ est impair} \\ & & \frac{n+4}{2} \text{ multiplications } /n \text{ additions} \quad \text{si } n \text{ est pair} \end{cases}$$

On peut alors observer que l'évaluation d'un polynôme impair de degré $n \geq 5$ avec Knuth & Eve nécessite moins d'opérations qu'avec Horner (4 multiplications au lieu de 5 avec Horner pour $n = 5$). Sinon, l'évaluation d'un polynôme pair de degré $n \geq 6$ avec Knuth & Eve nécessite également moins d'opérations qu'avec Horner (5 multiplications au lieu de 6 avec Horner pour $n = 6$).

2.5 Analyse des erreurs d'arrondi

On considère un polynôme a de degré 3. Son schéma d'évaluation est : $(a_3 y + a_2)(w - \alpha_1) + \beta_0$, avec $y = x + c$ et $w = y^2$. Son évaluation dans le modèle flottant est :

1. $\widehat{y} = (x + c)\langle 1\rangle$ et $\widehat{w} = (\widehat{y} \times \widehat{y})\langle 1\rangle = (x + c)^2\langle 3\rangle = y^2\langle 3\rangle$
2. $\widehat{z} = a_3\widehat{y}\langle 2\rangle + a_4\langle 1\rangle = a_3(x + c)\langle 3\rangle + a_4\langle 1\rangle$
3. $\widehat{r}_1 = (\widehat{w} - \alpha_1)\langle 1\rangle = ((x + c)\langle 4\rangle - \alpha_1\langle 1\rangle) = ((x + c) - \alpha_1\langle 3\rangle)\langle 4\rangle$
4. $\widehat{r}_2 = (\widehat{z} \times \widehat{r}_1)\langle 1\rangle = (a_3(x + c)\langle 8\rangle + a_4\langle 6\rangle)((x + c) - \alpha_1\langle 3\rangle)$
5. $\widehat{r}_3 = (\widehat{r}_2 + \beta_0)\langle 1\rangle = (a_3(x + c)\langle 9\rangle + a_4\langle 7\rangle)((x + c) - \alpha_1\langle 3\rangle) + \beta_0\langle 1\rangle$

L'erreur absolue est alors définie par : $|r_3 - a(x)| \leq \gamma_9 \cdot \tilde{a}(|x|)$, avec $\tilde{a}(|x|)$ le polynôme a sous la forme de Knuth & Eve.

Plus généralement, on peut conclure en majorant les erreurs absolue et relative de la manière suivante :

$$|\widehat{r} - a(x)| \leq \gamma_{3n}\tilde{a}(|x|) \quad \text{et} \quad \frac{|\widehat{r} - a(x)|}{|a(x)|} \leq \gamma_{3n}\frac{\tilde{a}(|x|)}{|a(x)|}.$$

On a considéré que le décalage n'était pas nul. Si le décalage est nul, les erreurs absolue et relative sont majorées de la manière suivante :

$$|\widehat{r} - a(x)| \leq \gamma_{2n}\tilde{a}(|x|) \quad \text{et} \quad \frac{|\widehat{r} - a(x)|}{|a(x)|} \leq \gamma_{2n}\frac{\tilde{a}(|x|)}{|a(x)|},$$

c'est à dire les mêmes que Horner.

2.6 Résultats numériques et observations

2.6.1 Similitude avec Horner

Comme on l'a déjà observé, cet algorithme se comporte comme la méthode de Horner dans le sens où, une fois que on a calculé $a_{2m+1}(q(y) + x)$, le schéma d'évaluation est un schéma séquentiel.

$res \leftarrow a_{2m+1}(q(y) + x)$;
pour i **de** $m - 1$ **à** 0 **faire**
  $res \leftarrow res \times (y - \alpha_{i+1}) + \beta_i$;
fin pour

Deux points font que cet algorithme reste toutefois plus intéressant que Horner. Tout d'abord, le nombre d'additions est le même que pour Horner, mais il nécessite environ deux fois moins de multiplications. D'autre part, tous les coefficients $y-\alpha_{i+1}$ peuvent être évalués indépendamment.

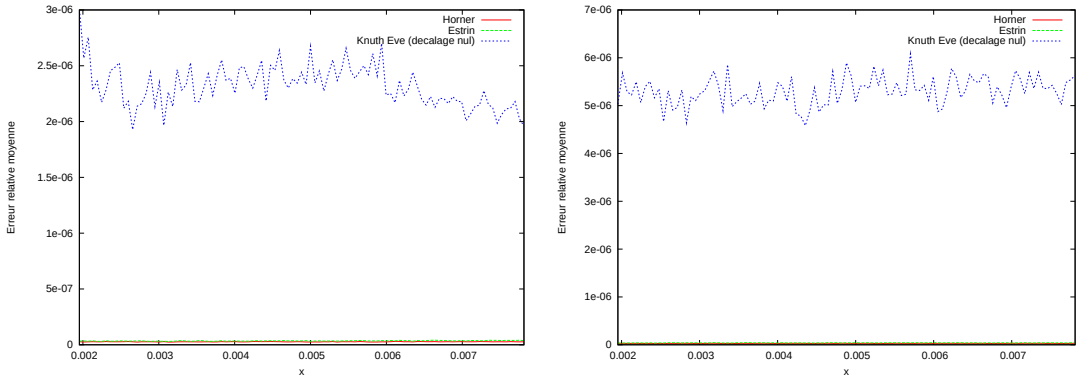
2.6.2 Effet du décalage

On avait l'intuition que le décalage pouvait avoir un effet néfaste sur l'erreur relative maximale lors de l'évaluation d'un polynôme par la méthode de Knuth & Eve. Cette première intuition a été confirmée par l'analyse d'erreur.

Cas du décalage nul. En pratique, on a évalué sur l'intervalle $[2^{-9}, 2^{-7}]$ des polynômes de degrés variés (7,12,15) dont l'évaluation ne nécessite pas de décalage. De manière générale, on peut observer que l'erreur relative par la méthode de Knuth & Eve est légèrement plus élevée que celle obtenue par la méthode de Horner (Fig. 2.1).

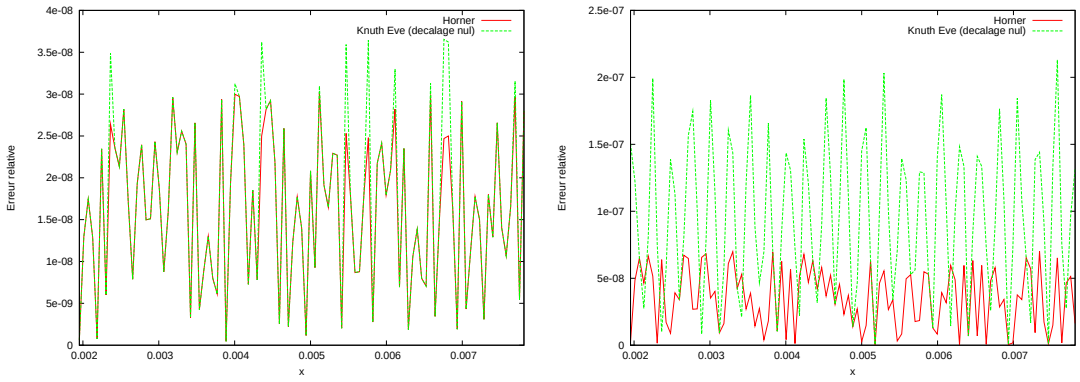
Cependant, dans certains cas, si le décalage est nul, l'erreur relative obtenue par la méthode de Knuth & Eve peut être comparable à celle obtenue avec Horner, voire à peine plus faible.

La figure 2.2 présente l'erreur relative de l'évaluation de polynômes de degré 7 et 15 par la méthode de Knuth & Eve. Dans ces deux cas, l'erreur relative est comparable à celle de Horner mais reste plus faible ($\approx 3.6 \times 10^{-8}$ pour le polynôme de degré 7 au lieu de $\approx 3.0 \times 10^{-8}$ avec



(a) polynômes de degré 7 (b) polynômes de degré 15

FIG. 2.1 – Erreurs relatives moyennes pour la méthode de Knuth & Eve



(a) polynôme de degré 7 (b) polynôme de degré 15

FIG. 2.2 – Erreurs relatives pour la méthode de Knuth & Eve

Horner, et $\approx 2.1 \times 10^{-7}$ au lieu de $\approx 7.0 \times 10^{-8}$ avec Horner pour le polynôme de degré 15). Finalement, la figure 2.3 montre la courbe d'erreur relative pour l'évaluation d'un polynôme de degré 12 pour laquelle l'erreur relative maximale pour la méthode de Knuth & Eve est inférieure à celle pour la méthode de Horner.
On peut malheureusement observer que parfois même sans décalage, l'erreur relative peut être élevée par rapport à l'erreur relative fournie par la méthode de Horner.

Cas du décalage non nul. En faisant augmenter le décalage, on peut s'apercevoir que l'erreur maximale moyenne augmente également. Et de manière plus générale, plus le décalage est élevé, plus l'erreur relative maximale est élevée. Ceci nous conforte quelque peu dans notre première intuition.

La figure 2.4 illustre l'effet du décalage sur l'erreur relative maximale par l'évaluation par la méthode de Knuth & Eve. En plus, du fait que plus le décalage est faible, plus l'erreur relative maximale le sera également, on peut observer que, de manière générale, les décalages déterminés par l'algorithme (méthodes de Eve et Knuth) sont pessimistes. Il se peut qu'entre ces deux bornes il existe un décalage qui conduise à une précision meilleure, mais rien ne le garantit (Fig 2.5). Ce sont les premiers qui garantissent le bon déroulement de l'algorithme.

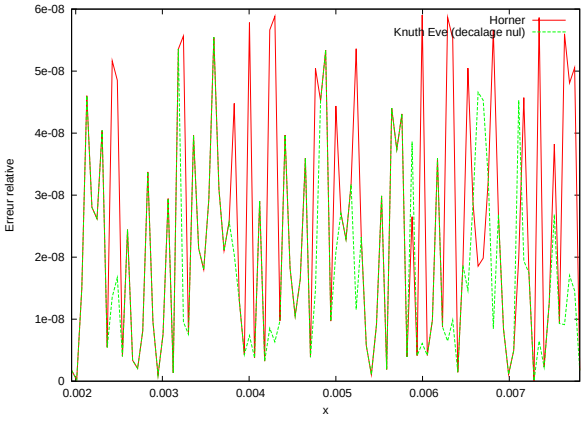
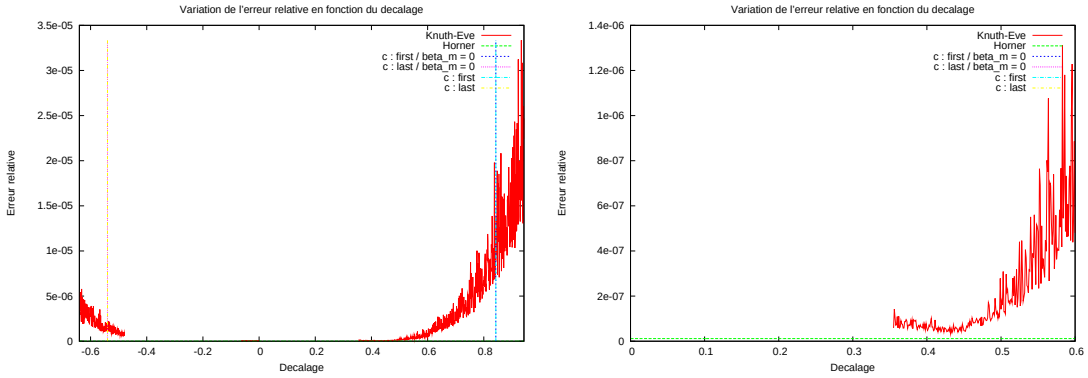


FIG. 2.3 – Cas où l'erreur relative pour Knuth & Eve est inférieure à celle pour Horner



(a) (b)

FIG. 2.4 – Effets du décalage sur l'erreur relative maximale pour Knuth & Eve

On peut cependant observer que l'effet du décalage peut parfois rétablir une certaine précision, l'erreur relative maximale étant alors comparable à celle par la méthode de Horner. Sur la figure 2.4(b), on peut observer que pour un décalage entre approximativement 3.5×10^{-1} et 4.3×10^{-1}, l'erreur relative diminue, alors que le décalage augmente autrement. Le décalage a alors un effet positif sur l'erreur relative, et une étude plus fine montre que pour un décalage de $\approx 4.33 \times 10^{-1}$ l'erreur relative maximale sera de $\approx 3.2 \times 10^{-8}$ contre $\approx 3.1 \times 10^{-8}$ par la méthode de Horner.

Cas du décalage pour lequel $\beta_0 = 0$. Knuth a remarqué que l'algorithme peut déterminer un décalage de telle sorte que le coefficient β_0 soit nul, et ce dans le but de diminuer de 1 le nombre total d'additions. Mais qu'en est-il, dans ce cas là, de la précision ?

Intuitivement, on peut dire que l'utilisation de ces décalages entraîne une erreur relative plus élevée. En effet, on a pu remarquer que les décalages pour lesquels $\beta_0 = 0$ sont des décalages « pessimistes », et que parfois il existe des décalages inférieurs pour lesquels l'algorithme fonctionne également. On peut par ailleurs remarquer que le décalage d'après la méthode de Eve (pour lequel a a au moins $n-1$ racines à partie réelle non négatives ou non positives) peut être inférieur (ou le même) que celui pour lequel $\beta_0 = 0$.
La pratique confime cette intuition (Fig. 2.6). On peut effectivement observer que l'évaluation par Knuth & Eve avec $\beta_0 = 0$ fournit une erreur relative moyenne supérieure à celle fournie par la méthode de Horner.

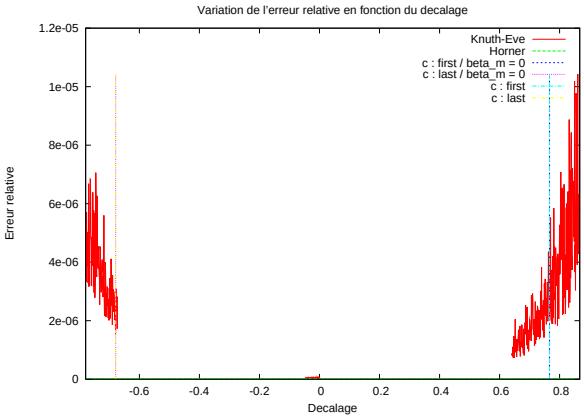


FIG. 2.5 – Cas où certains décalages ne fonctionnent pas forcément entre ceux déterminés par Knuth & Eve

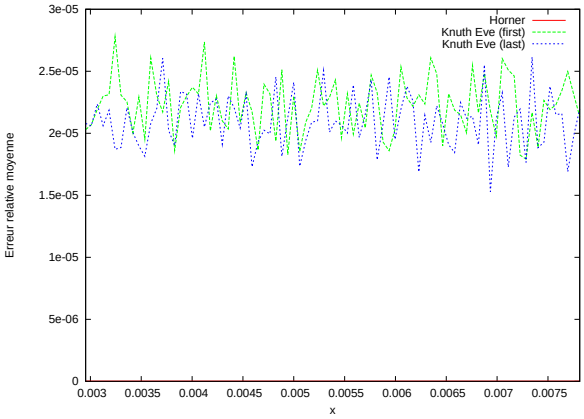


FIG. 2.6 – Erreur relative moyenne pour l'évaluation par Knuth & Eve avec $\beta_0 = 0$

2.6.3 Effet de la permutation

Les β_i dépendent directement des α_i, et suivant l'ordre dans lequel les α_i (racines de h) sont traités, les β_i déterminés peuvent être différents. On appelle *permutation* une permutation possible de α_i. Deux observations peuvent être effectuées.

Premièrement, pour un polynôme donné, on peut observer, comme le prévoyait Knuth [13], que l'erreur relative varie suivant la permutation utilisée. Ce phénomène est illustré par la figure 2.7 qui montre la courbe d'erreur relative pour l'évaluation d'un polynôme de degré 7 sur l'intervalle $[2^{-9}, 2^{-7}]$, par la méthode de Knuth & Eve pour chaque permutation de α_i possible (ici, [1,2,3] désigne la permutation $[\alpha_1, \alpha_2, \alpha_3]$).

Autrement, une permutation qui minimise l'erreur relative pour l'évaluation d'un polynôme en une valeur de x donnée, ne la minimise pas forcément pour toute valeur de x d'un intervalle. Lorsque on veut une « meilleure »permutation, on doit considérer celle qui minimise l'erreur relative maximale.

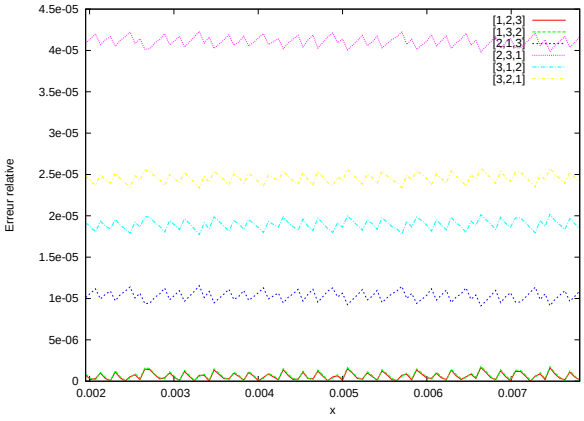


FIG. 2.7 – Effet de la permutation sur l'erreur relative pour Knuth & Eve

2.6.4 Algorithmes hybrides

On a observé que le décalage et la permutation interviennent dans la stabilité numérique de l'algorithme. Il est alors délicat de déterminer la meilleure combinaison décalage / permutation. On a observé que sous certaines conditions, l'erreur relative maximale obtenue par la méthode de Knuth & Eve peut être plus élevée que celle obtenue par la méthode de Horner.

Pour pallier à ce manque de précision, on propose d'utiliser des schémas d'évaluation hybrides. On considère un polynôme a de degré n : $a(x) = \sum_{i=0}^{n} a_i x^i$. On peut alors évaluer le polynôme a', partie haute du polynôme a, de degré $n-2$ avec la méthode de Knuth & Eve, qui peut être moins précise que Horner, mais qui est beaucoup plus rapide, puis terminer l'évaluation par la méthode de Horner (ou celle de Estrin).

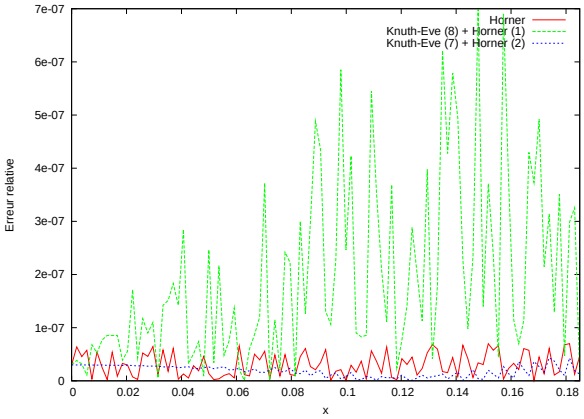


FIG. 2.8 – Algorithme hybride pour un polynôme de degré 9

En pratique, il s'avère que ce genre de schéma est suffisamment précis et permet de compenser avec Horner ou Estrin la perte en précision due à l'algorithme utilisé sur le haut du polynôme. Sur la figure 2.8, on peut voir trois courbes. La première correspond à la courbe d'erreur relative obtenue par la méthode de Horner pour l'évaluation d'un polynôme de degré 9 sur l'intervalle $[0, 0.1849]$. En pratique, appliquer Knuth & Eve au polynôme de degré 9 fournit de très mauvais

résultats. On évalue donc la partie haute du polynôme de degré 8 avec Knuth & Eve, puis on termine avec une itération de Horner (courbe 2). On fait de même avec une partie haute de degré 7 et deux itérations de Horner. Cette méthode permet de compenser la perte de précision due à Knuth & Eve.

Chapitre 3

Algorithme de Paterson & Stockmeyer

3.1 Description

L'algorithme de Paterson & Stockmeyer [19] est un algorithme permettant d'évaluer des polynômes de degré n en $\frac{n}{2} + 2\log(n)$ multiplications. Il utilise un préconditionnement rationnel. Cependant, pour pouvoir être évalué tel quel par cet algorithme, le polynôme doit être unitaire (coefficient de plus haut degré égal à 1) et de degré de la forme $n = 2^p - 1$.

Si le polynôme évalué n'est pas unitaire, il sera ramené au polynôme unitaire correspondant (par division par le coefficient de plus haut degré), et l'évaluation nécessitera une multiplication supplémentaire pour revenir au polynôme initial. Autrement, si le polynôme est de degré $n \neq 2^p - 1$, il sera découpé en polynômes de degré $2^i - 1$ selon l'écriture binaire de n, qui seront évalués séparément par l'algorithme de Paterson & Stockmeyer. Ensuite, $\log(n)$ multiplications seront nécessaires pour déterminer le résultat final.

Pour simplifier la présentation, on considérera des polynômes unitaires de degré $n = 2^p - 1$.

3.2 Intérêt principal de cette approche

On a vu que la méthode de Estrin permet l'évaluation d'un polynôme en $n + \log(n)$ multiplications et n additions, en utilisant la stratégie « diviser pour régner ». La figure 1.1 montre que son schéma d'évaluation peut être représenté sous la forme d'un arbre binaire. L'évaluation de chacune des feuilles nécessite une multiplication et une addition.

L'idée est ici d'utiliser la même stratégie, mais en utilisant uniquement des polynômes unitaires, jusqu'aux feuilles, afin notamment de supprimer la multiplication dans chacune des feuilles de l'arbre. Cette stratégie ne peut être effectuée sans un certain décalage.

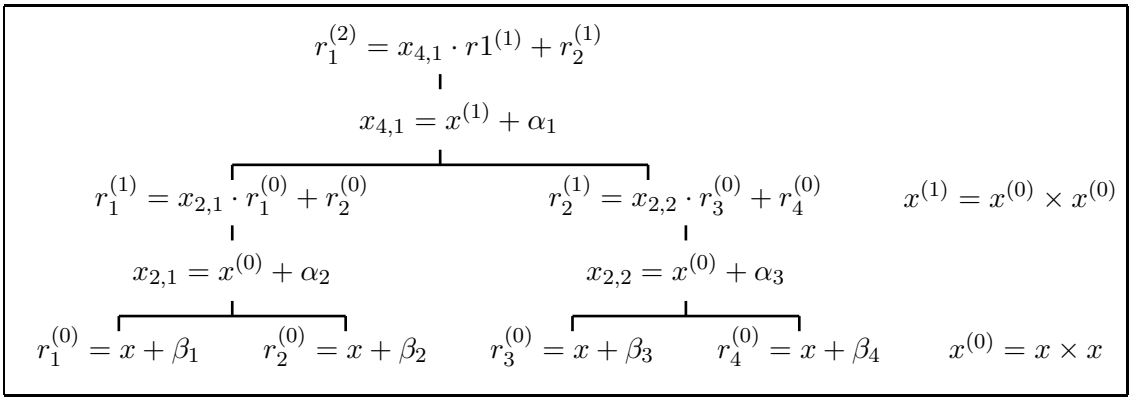


FIG. 3.1 – Méthode de Paterson & Stockmeyer en degré $n = 7$

3.3 Schéma d'évaluation et préconditionnement

On considère un polynôme a de degré n : $a(x) = x^n + \sum_{i=0}^{n-1} a_i x^i$, avec $n = 2^p - 1$. Ce polynôme est alors évalué avec l'algorithme de Paterson & Stockmeyer de la manière suivante :

$$a(x) = (x^{\frac{n+1}{2}} + \alpha)(x^{\frac{n-1}{2}} + \sum_{i=0}^{\frac{n-3}{2}} a_{i+\frac{n+1}{2}} x^i) + (x^{\frac{n-1}{2}} + \sum_{i=0}^{\frac{n-3}{2}} b_i x^i),$$

avec $\alpha = a_{p-1} - 1$ et $b_i = a_i - \alpha \cdot a_{p+i}$. Les polynômes $(x^{\frac{n-1}{2}} + \sum_{i=0}^{\frac{n-3}{2}} a_{i+\frac{n+1}{2}} x^i)$ et $(x^{\frac{n-1}{2}} + \sum_{i=0}^{\frac{n-3}{2}} b_i x^i)$ seront également évalués avec Paterson & Stockmeyer.

Par exemple, pour un polynôme $a(x) = x^3 + a_2 x^2 + a_1 x + a_0$ de degré $n = 3$, l'algorithme évalue la formule suivante :

$$a(x) = \big(x^2 + (a_1 - 1)\big) \cdot \big(x + a_2\big) + \big(x + (a_0 - (a_1 - 1) \cdot a_2)\big).$$

Pour cet algorithme, le préconditionnement est rationnel (contrairement à celui de la méthode de Knuth & Eve) et on peut donc le faire de manière exacte sur les rationnels. Donc, les rationnels α_i et β_i sont calculés exactement en Maple.

Il est à noter que, comme Estrin, le schéma d'évaluation peut être représenté sous la forme d'un arbre binaire (Fig. 3.1). Le préconditionnement consiste alors en une descente dans l'arbre pour la détermination des coefficients, et l'évaluation en une remontée de l'arbre.

3.4 Analyse de complexité

En notant $C(p)$ le coût de l'évaluation d'un polynôme de degré $n = 2^p - 1$, on a, en sachant que l'on connait les puissances de x $(x^2, x^4, \ldots, x^{2^{p-1}})$:

$$\begin{cases} C(1) &= & 1 \text{ addition} \\ C(p) &= & 2C(p-1) + 1 \text{ multiplication } + 2 \text{ additions} \end{cases}$$

Plus généralement, on peut observer que $C(p) = \frac{n+1}{2} - 1$ multiplications, et $\frac{3n-1}{2}$ additions. On a, ici, considéré que le polynôme à évaluer était unitaire. Si cela n'est pas le cas, une multiplication finale est nécessaire à l'évaluation pour revenir au polynôme initial, ce qui fait au plus

$\frac{n+1}{2}$ multiplications. Finalement, $\log(n+1)-1$ multiplications sont encore nécessaires pour le calcul des puissances de x (x^2,x^4,...,$x^{2^{p-1}}$).

L'évaluation d'un polynôme non unitaire par la méthode de Paterson & Stockmeyer utilise $\frac{n+1}{2}+\log(n+1)-1$ multiplications et $\frac{3n-1}{2}$ additions. Par exemple, pour $n=15$, l'algorithme utilisera au plus 11 multiplications et 22 additions, et pour $n=31$, 20 multiplications et 46 additions. On a certes des additions supplémentaires par rapport à Horner, mais on fait chuter le nombre de multiplications, notamment pour les degrés élevés. Cette stratégie est particulièrement intéressante dès que la multiplication est plus coûteuse que l'addition, comme cela l'était auparavant, ou bien comme cela l'est actuellement sur des formats hybrides comme le format double-double. Mais également, comme Estrin, il est plus parallèle que Horner.

3.5 Analyse des erreurs d'arrondi

On considère un polynôme a de degré 3 : $a(x)=a_3x^3+a_2x^2+a_1x+a_0$. Son schéma d'évaluation avec la méthode de Paterson & Stockmeyer est le suivant :

$$a_3\cdot\big((x^2+c)\cdot(x+a)+(x+b)\big),$$

les coefficients c, a et b étant calculés exactement puis arrondis au plus près lors du préconditionnement. Déroulons maintenant ce schéma d'évaluation pour analyser l'effet des erreurs d'arrondi dans le modèle flottant standard :

1.	$\widehat{r}_1=(x+a)\langle1\rangle=x\langle1\rangle+a\langle1\rangle$
2.	$\widehat{r}_2=(x+b)\langle1\rangle=x\langle1\rangle+b\langle1\rangle$
3.	$\widehat{r}_3=(x\times x\langle1\rangle+c)\langle1\rangle=x^2\langle2\rangle+c\langle1\rangle$
4.	$\widehat{r}_4=(\widehat{r}_3\widehat{r}_1\langle1\rangle+\widehat{r}_2)\langle1\rangle=x^3\langle5\rangle+ax^2\langle5\rangle+cx\langle2\rangle+bx\langle2\rangle+ca\langle2\rangle+b\langle2\rangle$
5.	$\widehat{r}_5=(a_3\widehat{r}_4)\langle1\rangle=a_3x^3\langle6\rangle+a_3ax^2\langle6\rangle+a_3cx\langle3\rangle+a_3bx\langle3\rangle+a_3ca\langle3\rangle+a_3b\langle3\rangle$

Il est à noter que $\widehat{r}_5$ est la valeur effectivement calculée pour $a(x)$. On obtient :

$$\begin{aligned}\widehat{r}_5 &= a_3x^3\langle6\rangle+a_3ax^2\langle6\rangle+a_3cx\langle3\rangle+a_3bx\langle3\rangle+a_3ca\langle3\rangle+a_3b\langle3\rangle\\ &= a_3x^3(1+\theta_6)+a_3ax^2(1+\theta_6)+a_3cx(1+\theta_3)+a_3bx(1+\theta_3)+a_3ca(1+\theta_3)+a_3b(1+\theta_3)\end{aligned}$$

Rappelons que, pour tout i, $|\theta_i|\le\gamma_6$. Donc, on obtient finalement pour l'erreur absolue :

$$\begin{aligned}|\widehat{a}-a(x)| &\le |a_3x^3\cdot\theta_6|+|a_3ax^2\cdot\theta_6|+|a_3cx\cdot\theta_3|+|a_3bx\cdot\theta_3|+|a_3ca\cdot\theta_3|+|a_3b\cdot\theta_3|\\ &\le \gamma_6(|a_3x^3|+|a_3ax^2|+|a_3cx|+|a_3bx|+|a_3ca|+|a_3b|)\\ &\le \gamma_6.\tilde{a}(|x|),\end{aligned}$$

où $\tilde{a}(|x|)$ est le polynôme sous la forme de Paterson & Stockmeyer avec les coefficients α_i et β_j en valeur absolue. De manière plus générale, on peut majorer les erreurs relative et absolue de la manière suivante :

$$|\widehat{r}-a(x)|\le\gamma_{n+\log(n+1)}\tilde{a}(|x|)\quad\text{et}\quad\frac{|\widehat{r}-a(x)|}{|a(x)|}\le\gamma_{n+\log(n+1)}\frac{\tilde{a}(|x|)}{|a(x)|}.$$

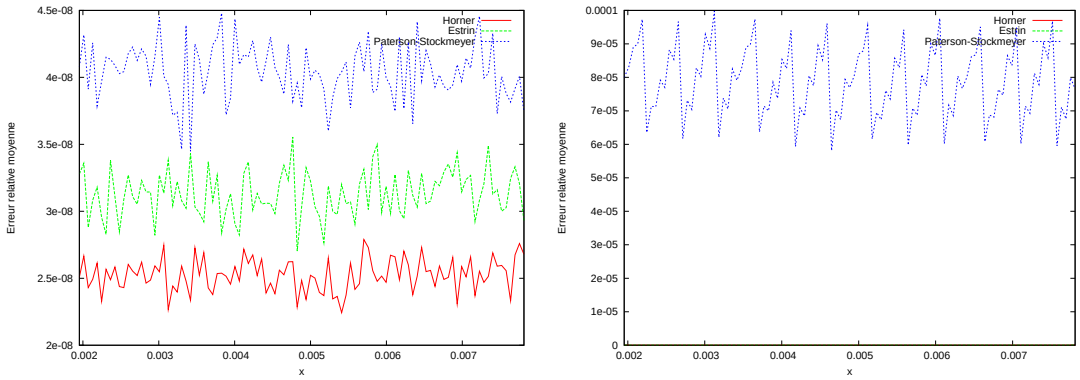
On peut observer que si $\alpha_i\ge0$ pour tout i, si $\beta_j\ge0$ pour tout j, et si $x\ge0$, alors $\tilde{a}(|x|)/|a(x)|=1$. Cependant, il ne paraît pas évident a priori d'en déduire des conditions suffisantes sur les coefficients a_i de départ.

3.6 Résultats numériques et observations

3.6.1 Similitude avec Estrin

On a pu observer que ce schéma s'inspire considérablement de la méthode de Estrin en termes de structure. Mais qu'en est-il en termes de précision ?

On a généré de manière aléatoire un ensemble de polynômes de degré 7 dont le problème d'évaluation en x est aussi bien conditionné avec Paterson & Stockmeyer que avec Horner et Estrin (ensemble 1), et on l'a comparé à un ensemble de polynômes de degré 7 tirés aléatoirement (ensemble 2). La figure 3.2 montre les erreurs relatives moyennes pour l'évaluation par Paterson & Stockmeyer sur ces deux jeux de 200 polynômes de degré 7.



(a) ensemble 1 (b) ensemble 2

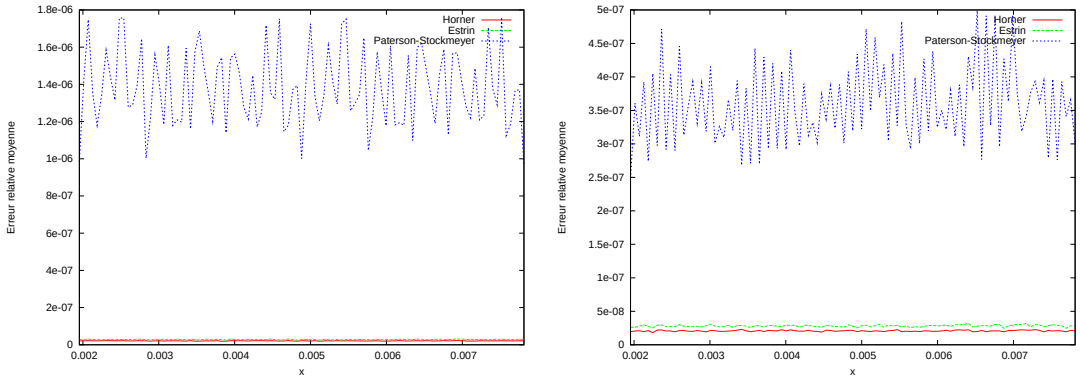
FIG. 3.2 – Erreurs relatives moyennes pour l'évaluation par Horner, Estrin et Paterson & Stockmeyer

On peut alors remarquer que lorsque le problème d'évaluation est bien conditionné (ensemble 1), la méthode de Paterson & Stockmeyer fournit une erreur relative du même ordre de grandeur que Horner ou Estrin, tout en restant un peu plus élevée.

Cependant, on a pu remarquer que pour des polynômes de degré élevé (15, 31), l'erreur relative moyenne était beaucoup plus élevée. Cela vient peut être du fait que plus le polynôme a un degré élevé, plus le nombre de décalages sera important et plus les décalages successifs auront un impact sur les coefficients α_i et β_j. Le problème de l'évaluation du polynôme obtenu peut alors être mal conditionné, ce qui peut engendrer une erreur relative importante.

3.6.2 Faut-il que les polynômes soient unitaires ?

On a observé que le polynôme, pour pouvoir être évalué par la méthode de Paterson & Stockmeyer, devait être unitaire. On a évalué 200 polynômes unitaires et 200 non unitaires sur l'intervalle $[2^{-9}, 2^{-7}]$. On obtient les courbes de la figure 3.3.
On peut remarquer que, sur ces figures, l'erreur relative est plus faible pour l'évaluation de polynômes unitaires que celle pour les polynômes non unitaires. Ce n'est pas si étonnant puisque lors de l'évaluation de polynômes non unitaires, on divise dans un premier temps tous les coefficients du polynôme, en introduisant une première erreur, puis on multiplie le résultat obtenu par le coefficient de plus haut degré du polynôme initial, introduisant une nouvelle erreur. L'évaluation de polynômes unitaires ne fait pas intervenir ces opérations, et par conséquent diminue l'erreur.



(a) non unitaires (b) unitaires

FIG. 3.3 – Erreurs relatives moyennes pour l'évaluation Paterson & Stockmeyer

3.6.3 Évaluation en le polynôme réciproque

Soit $a(x) = \sum_{i=0}^{n} a_i x^i$. On a pu remarquer que si le polynôme est un polynôme « décroissant », c'est à dire si $a_0 > a_1 > \ldots > a_n$, et qu'il décroît très rapidement, l'évaluation par l'algorithme de Paterson & Stockmeyer sera très peu précise. En effet, si le coefficient a_n est beaucoup plus faible que le coefficient a_0, lorsque ce polynôme sera ramené au polynôme unitaire, chaque coefficient sera multiplié par $\frac{1}{a_n}$, qui lui est très grand. Si les coefficients de degrés faibles sont grands, on pourra alors avoir un dépassement de capacité, le quotient ne pourra alors pas être représenté en machine dans le format considéré. Alors, l'évaluation du polynôme ne pourra pas être effectuée. On s'est aperçu que pour remedier à cela, on peut utiliser le polynôme réciproque de $a(x)$, c'est à dire $b(x) = x^n \times a(x^{-1})$.

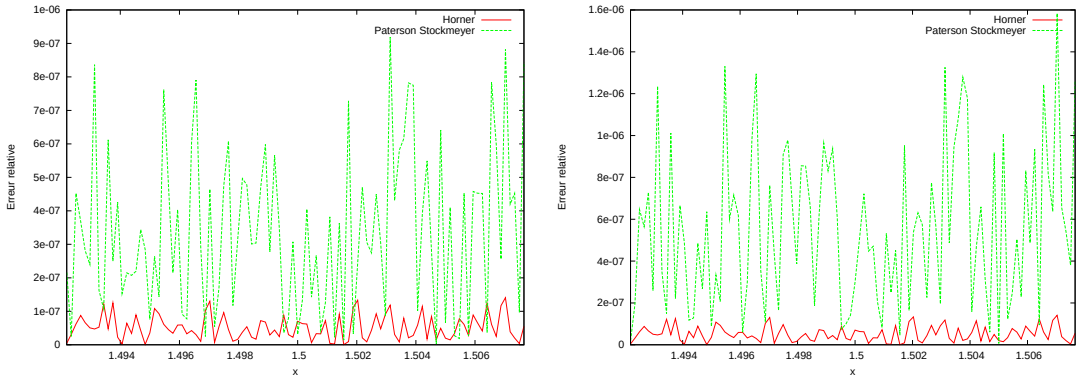
L'évaluation de a se fera de la manière suivante : on évalue $y = x^{-1}$, puis $b(y)$ avec l'algorithme de Paterson & Stockmeyer, et enfin, on reconstruit $a(x)$ via $a(x) = x^n \times b(y)$. Le polynôme réciproque pourra être déterminé pendant le préconditionnement, mais l'évaluation nécessitera 5 multiplications supplémentaires par rapport au coût de Paterson & Stockmeyer seul.

Les développements de Taylor de la plupart des fonctions élémentaires usuelles ont cette structure décroissante. Et il s'avère, en pratique, que cette méthode est satisfaisante. Par exemple, on considère les développements en séries de Taylor de la fonction $\cos(x)$, aux ordres 16 et 32. Les polynômes en x^2 associés, de degré respectif 7 et 15, ont un coefficient de degré 0 égal à 1, et leurs coefficients décroissent rapidement. On a pu vérifier que l'évaluation échouait si l'on appliquait Paterson & Stockmeyer directement sur ces polynômes.
Par contre, si l'on passe par l'évaluation du polynôme réciproque, on obtient des résultats, en termes d'erreur relative, comparables à l'erreur de Horner (Fig. 3.4).

3.6.4 Algorithmes hybrides

Comme on a déjà pu l'observer pour l'algorithme de Knuth & Eve, l'utilisation d'un schéma hybride avec une autre méthode, notamment avec Horner ou Estrin, permet de compenser la perte de précision due à l'utilisation de l'algorithme de Paterson & Stockmeyer. C'est notamment comme cela que l'on a implanté le logarithme népérien dans CR-Libm.

Cette méthode sera d'autant plus utile si le polynôme à évaluer est d'un degré $n \neq 2^p - 1$.



(a) polynôme de degré 7

(b) polynôme de degré 15

FIG. 3.4 – Erreurs relatives pour l'évaluation par la méthode du polynôme réciproque

Si on évalue un polynôme de degré $n > 2^p - 1$, on évaluera le haut du polynôme de degré $2^p - 1$ avec Paterson & Stockmeyer, puis le bas avec une autre méthode (Horner ou Estrin). Soit alors p maximum tel que $m = 2^p - 1 < n$ et soit $a(x) = a_h(x) \times x^{n-m} + a_l(x)$, avec $a_h(x)$ de degré m et $a_l(x)$ de degré $< n - m$. Alors, on évaluera la « partie haute » $a_h(x)$ avec Paterson & Stockmeyer puis la « partie basse » $a_l(x)$ avec Horner ou Estrin.

Chapitre 4

Implantation dans CR-Libm

4.1 Fonctionnement de CR-Libm

4.1.1 Processus d'implantation d'une fonction élémentaire

Dans CR-Libm, le processus d'implantation suit trois étapes.

Approximation polynomiale. On considère une fonction f à évaluer sur un intervalle I. La première étape consiste à déterminer un polynôme d'approximation a sur cet intervalle pour cette fonction, avec la routine `minimax` du package `numapprox` de Maple ou l'algorithme de Remez, par exemple. Le polynôme obtenu approche au mieux f sur I. Ensuite, les coefficients du polynôme sont arrondis au plus près dans le format considéré pour pouvoir être représentables en machine. On introduit alors un premier type d'erreur : l'erreur d'approximation. Ceci est dû au fait que le polynôme n'approche pas exactement la fonction f. En effet, a avec ces coefficients non arrondis est juste un bon approximant de la fonction f obtenu sur I.

Implantation et test d'évaluation. La deuxième étape consiste à évaluer le polynôme précédemment obtenu avec une ou plusieurs des méthodes d'évaluation présentées. Comme on a pu l'observer, on introduit dans cette étape un deuxième type d'erreur : l'erreur d'évaluation ou erreur d'arrondi.

Preuve de la borne d'erreur. Une fois qu'on a trouvé un schéma d'évaluation valable pour le polynôme a, on doit, en troisième étape, prouver que l'utilisation de ce schéma sur a fournira une erreur relative inférieure à une certaine borne. À ce stade, on doit prendre en compte à la fois l'erreur d'approximation et les erreurs d'arrondi. Cette preuve est effectuée avec le générateur automatique de preuves Gappa [2].

4.1.2 Évaluation d'une fonction

L'évaluation d'une fonction dans CR-Libm, plus particulièrement du polynôme qui l'approche, se fait en deux étapes. Une première étape rapide permet de déterminer rapidement un premier résultat avec une erreur relative inférieure à 2^{-60}, ce qui est largement suffisant pour ensuite arrondir correctement le résultat sur 53 bits (format double précision). Si cette première étape échoue, la seconde étape est une étape plus lente, qui évalue un polynôme de plus grand degré, et fournit un résultat avec une plus grande précision, permettant de décider du résultat pour les cas non résolus dans la phase rapide. On voit alors que le temps total d'évaluation d'une fonction est :

$$T = T_{rapide} + p \cdot T_{lente}.$$

où p est la probabilité de lancer la phase lente. Il s'avère que cette probabilité est proportionnelle à la précision de la phase rapide : $T = T_{rapide} + 2^{-60} \cdot T_{lente}$.

On s'était fixé pour objectif, afin de tester les algorithmes efficaces étudiés, d'accélérer l'évaluation complète de fonctions élémentaires. On s'est donc restreint à accélérer la phase rapide, tout en garantissant la même précision que celle actuelle.

4.2 Algorithme hybride pour le logarithme népérien

Le premier polynôme sur lequel on a implanté ces algorithmes efficaces a été le polynôme de la phase rapide de la fonction logarithme népérien, $\ln(x)$, de CR-Libm.

4.2.1 Comment est implanté le logarithme népérien ?

En phase rapide, le logarithme népérien est approché par un polynôme de degré 7. Le coefficient de degré 1 est égal à 1. Le coefficient de degré 0 est une valeur lue dans une table au cours de l'évaluation, dépendant de x [3]. Ce polynôme est ensuite évalué sur l'intervalle $[-2^{-7}, 2^{-7}]$.

4.2.2 Solution proposée

Schéma d'évaluation. Actuellement, ce polynôme est évalué dans CR-Libm avec l'algorithme de Estrin. Son coefficient de degré 1 étant égal à 1, cette évaluation nécessite 8 multiplications et 7 additions. On a testé différents algorithmes et schémas d'évaluation. Il s'est avéré qu'un d'entre eux permettait à la fois d'être aussi précis que celui de Estrin, mais également plus rapide. Il consiste alors à utiliser un schéma hybride Paterson & Stockmeyer/Estrin.

L'idée est alors d'évaluer le haut du polynôme par la méthode de Paterson & Stockmeyer, et le bas par la méthode de Estrin, et finalement de déterminer le résultat final à partir des deux résultats intermédiaires calculés indépendamment. On considère le polynôme a de degré 7. Son schéma d'évaluation par la méthode de Paterson & Stockmeyer/Estrin est le suivant :

$$a(x) = \big((x^2+\alpha)(x+\beta_1) + (x+\beta_2)\big)x^4 \times a_7 + \big((c_3x + c_2)x^2 + (x + logirh)\big)$$

avec $a'_i = \frac{a_i}{a_7}$, et $\alpha = a'_5 - 1$, $\beta_1 = a'_6$ et $\beta_2 = a'_4 - ca'_6$. Le préconditionnement consiste uniquement à calculer α, β_1 et β_2. La figure 4.1 montre la première courbe d'erreur relative obtenue en pratique avec ce shéma.

On peut s'apercevoir que ce schéma semble être aussi précis que Estrin et que Horner. Et en nombre d'opérations, il nécessite 7 multiplications et 8 additions, contrairement à Estrin qui nécessitait 8 multiplications et 7 additions.

Implantation et tests. Les tests de précision ont été effectués avec succès. En termes de performances, les tests ont été effectués sous environnement Linux sur un Pentium M 1.6 GHz, avec les compilateurs `gcc3.4` et `gcc4.0`. Le tableau 4.1 montre le nombre moyen de cycles, sur 10000 évaluations, en arrondi au plus près (mode d'arrondi par défaut) pour les deux méthodes. On peut s'apercevoir, qu'en arrondi au plus près, on gagne 5 cycles si le compilateur utilisé est `gcc3.4` et 3 cycles avec `gcc4.0`.

Preuve Gappa. Gappa est un générateur automatique de preuves. Il permet par arithmétique d'intervalle de prouver, notamment, la borne d'erreur sur un algorithme.

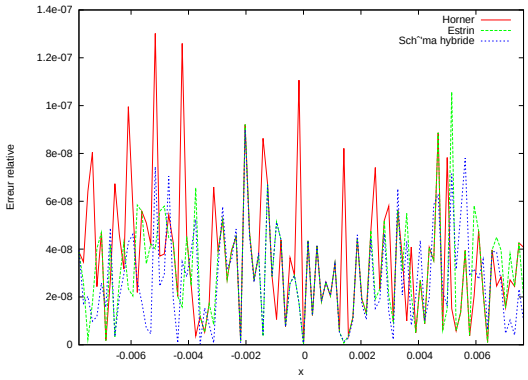


FIG. 4.1 – Erreur relative pour l'évaluation du logarithme

	CR-Libm	
	Estrin	PS/Estrin
`gcc-3.4`	143 cycles	138 cycles
`gcc-4.0`	151 cycles	148 cycles

TAB. 4.1 – Comparatif des schémas d'évaluation en nombre de cycles

On lui fournit le schéma d'évaluation en exact et dans le format considéré, la valeur des paramètres, et il retourne une borne sur l'erreur relative. Cette borne est prouvée correcte, et cette démarche sert de preuve. La preuve du logarithme avec cet outil, pour ce nouveau schéma d'évaluation, garantit une erreur relative inférieure à 2^{-62}, comme celle fournit pour la méthode de Estrin pure.

Cette méthode est donc aussi précise et plus rapide que celle précédemment utilisée (qui était déjà la plus rapide). Á terme, elle sera donc intégrée à CR-Libm pour implanter le logarithme.

4.3 Accélération apportée à la fonction $\arcsin(x)$

4.3.1 Comment est implantée la fonction $\arcsin(x)$ dans CR-Libm ?

La fonction $\arcsin(x)$ est une fonction impaire définie sur le domaine $[-1, 1]$. On a alors $\arcsin(-x) = \arcsin(x)$. La fonction est alors uniquement évaluée sur l'intervalle $[0, 1]$.

Contrairement à la fonction logarithme, $\arcsin(x)$ est approchée par dix polynômes. Plus particulièrement, le domaine d'évaluation est découpé en 10, et un polynôme d'approximation est utilisé sur chaque sous-domaine [4]. Le premier et le dernier polynôme sont de degré 9 et 17 respectivement. Les huit autres polynômes sont tous de degré 14. Ceci implique que dans CR-Libm, trois schémas d'évaluation uniquement sont utilisés.

4.3.2 Présentation des algorithmes utilisés

Schéma d'évaluation du premier polynôme. Le premier polynôme est un polynôme de degré 9 évalué sur l'intervalle $[0, 2^{-2.434403}]$. Jusqu'à maintenant, ce polynôme était évalué par la méthode de Horner. On a alors testé différents schémas d'évaluation hybrides, à partir des algorithmes étudiés. En terme de précision, la plupart d'entre eux était valable, mais en termes

de rapidité, le schéma retenu est un schéma hybride basé sur l'algorithme de Knuth & Eve et de Estrin.

On considère un polynôme de degré 9. Dans ce schéma, le haut du polynôme de degré 7 est évalué avec Knuth & Eve et le bas avec Estrin. Ensuite, deux multiplications et une addition sont nécessaires pour retourner le résultat final. De plus, afin de minimiser au maximum le nombre d'opérations, le polynôme de degré 7 est évalué par Knuth & Eve avec un décalage de telle sorte que β_0 soit nul. Le schéma d'évaluation est le suivant :

$$a_1(x) = \big((a_7(q_0 + x)(y - \alpha_3) + \beta_2)(y - \alpha_2) + \beta_1\big)(y - \alpha_1)x^2 + (a_1 x + a_0).$$

Ce schéma nécessite alors 7 multiplications et 8 additions, contrairement à 9 multiplications et 9 additions pour Horner.

Tests et preuve pour le premier polynôme. Comme pour le logarithme, la première phase de tests de précision a été effectuée avec succès. De plus, la preuve Gappa nous garantit une erreur relative inférieure à 2^{-60} (voire 2^{-101} si x est très proche de zéro).
En terme de performances, le tableau 4.2 montre le nombre de cycles moyen pour 10000 évaluations de $\arcsin(x)$ en arrondi au plus près avec Horner et avec ce nouveau schéma. On peut

	CR-Libm	
	Horner	KE/Estrin
`gcc-3.4`	342 cycles	276 cycles
`gcc-4.0`	342 cycles	283 cycles

TAB. 4.2 – Comparatif des schémas d'évaluation en nombre de cycles

remarquer qu'on a gagné, avec le compilateur `gcc3.4`, 66 cycles soit 19%.

Les autres polynômes de $\arcsin(x)$**.** L'évaluation des autres polynômes a été implantée avec la méthode de Estrin. A l'heure d'aujourd'hui, les premiers tests de précision ont été effectués avec succès, et des tests aléatoires plus complets ont été tournés pendant plus de six heures avec succès également. Il ne reste donc plus qu'à prouver ces algorithmes avec Gappa. Autrement, en termes de performances, l'utilisation de la méthode de Estrin a également diminué le temps d'évaluation. Le tableau 4.3 montre le nombre de cycles moyen pour 10000 évaluations de $\arcsin(x)$ en arrondi au plus près avec Horner et les trois nouveaux schémas implantés. On

	CR-Libm	
	Horner	KE/Estrin
`gcc-3.4`	342 cycles	259 cycles
`gcc-4.0`	342 cycles	269 cycles

TAB. 4.3 – Comparatif des schémas d'évaluation en nombre de cycles

pu finalement observer qu'avec l'utilisation des trois nouveaux schémas, on a gagné, avec le compilateur `gcc3.4`, 83 cycles soit environ 24%.

Conclusion et perspectives

Dans ce rapport, on a étudié deux méthodes d'évaluation à base de préconditionnement, la méthode de Knuth & Eve et la méthode de Paterson & Stockmeyer, et on les a comparées aux méthodes usuelles de Horner et de Estrin.

La méthode de Knuth & Eve permet d'évaluer un polynôme avec un schéma d'évaluation proche de celui de Horner. Après une étude de cet algorithme, on a pu observer l'importance du décalage dans le préconditionnement. On a dans un premier temps analysé de manière théorique que, si lors du préconditionnement on ne faisait pas de décalage du polynôme initial, la borne d'erreur de Knuth & Eve était la même que celle de Horner. Et en pratique, on a pu observer des cas où, sans décalage, l'évaluation du polynôme fournissait une erreur maximale comparable à celle de Horner, voire meilleure. Toujours à propos du décalage, Knuth et Eve ont proposé des méthodes permettant de le déterminer. Mais on a pu observer que les décalages déterminés par ces deux méthodes ne sont pas optimaux. En effet, ce sont des décalages qui garantissent effectivement le bon déroulement de l'algorithme, mais qui peuvent entraîner une augmentation des erreurs par rapport à celles pour la méthode de Horner. D'autres décalages plus faibles (ou plus élevés suivant le sens du décalage) peuvent être bien meilleurs.
De plus, dans la méthode de Knuth, le décalage obtenu est de telle sorte que le coefficient β_0 calculé par l'algorithme est nul. Certes, cette technique est intéressante du point de vue du nombre d'opérations (on gagne une addition) et par conséquent de la rapidité de l'algorithme, et elle a d'ailleurs été utilisée dans l'implantation de la fonction $\arcsin(x)$ de CR-Libm. Mais en ce qui concerne la précision, cette méthode n'est pas optimale et fournit généralement des erreurs relativement importantes. De manière générale, on a en fait pu observer que plus le décalage était faible, plus l'erreur relative était également faible.
Toujours à propos de ce premier algorithme, on a pu observer l'importance également du choix de la permutation des α_i. Knuth intuitait le fait que le choix de la permutation pouvait influer sur la précision de l'algorithme. En pratique, on a pu observer effectivement que suivant la permutation, pour un polynôme donné, l'erreur relative par Knuth & Eve pouvait varier considérablement.

La méthode de Paterson & Stockmeyer permet l'évaluation d'un polynôme avec un schéma d'évaluation inspiré de celui de Estrin. On a pu observer que si le problème d'évaluation d'un polynôme est bien conditionné avec Paterson & Stockmeyer, l'erreur relative est comparable à celle de Horner et Estrin. De plus, lorsque les polynômes sont unitaires, l'erreur relative moyenne est plus faible que dans le cas de l'évaluation de polynômes non unitaires.
Par ailleurs, on a pu observer que, lorsque le polynôme est décroissant et décroit très rapidement, cet algorithme peut devenir très imprécis. La méthode que l'on a utilisée pour résoudre ce problème consiste à évaluer le polynôme réciproque, puis à reconstruire le résulat final. Cette méthode fournit des résultats satisfaisants, au détriment de multiplications supplémentaires.

D'aute part, on a pu observer en pratique l'intérêt des schémas hybrides, comme celui utilisé dans l'implantation du logarithme ou bien du premier polynôme de $\arcsin(x)$. Ces schémas ont pour objectif d'évaluer le haut d'un polynôme avec une méthode rapide mais éventuellement peu précise, puis de « rattraper » l'erreur en terminant l'évaluation (partie basse) avec une méthode plus précise, comme Horner voire Estrin. En pratique, ces schémas fournissent un bon compromis précision/rapidité.

Pour finir, on a pu observer la difficulté, pour l'algorithme de Knuth & Eve, de déterminer la meilleure combinaison décalage/permutation. Il pourrait alors être intéressant d'automatiser ce processus. Ceci accélérerait considérablement le temps de conception d'un schéma d'évaluation.
Autrement, il pourrait également être intéressant d'étendre les travaux de N. Louvet sur le *schéma de Horner compensé* à la méthode de Knuth & Eve. Cette méthode permet de récupérer l'erreur effectuée au cours des calculs de la méthode de Horner, au détriment de quelques opérations supplémentaires. On pourrait alors observer dans quelle mesure on reste plus rapide que Horner malgré la compensation.
Finalement, il pourrait être intéressant de tester ces algorithmes sur des architectures spécifiques, comme notamment des architecture sans aucun parallèlisme ni pipeline pour la méthode de Knuth & Eve, ou bien justement sur des architectures vraiment parallèles pour la méthode de Paterson & Stockmeyer.

Bibliographie

[1] N. Brisebarre and J.-M. Muller. Correct rounding of algebraic functions. Article à paraître dans RAIRO Theoretical Informatics and Applications.

[2] F. de Dinechin, Ch. Q. Lauter, and G. Melquiond. Assisted verification of elementary functions. Technical Report 5683, INRIA, September 2005.

[3] F. de Dinechin, Ch. Q. Lauter, and J.-M. Muller. Fast and correctly rounded logarithms in double-precision. Technical Report RR2005-37, LIP, September 2005.

[4] C. Daramy-Loirat, D. Defour, F. de Dinechin, M. Gallet, N. Gast, Ch. Q. Lauter et J.-M. Muller. CR-Libm, A library of correctly rounded elementary functions in double-precision (documentation). Disponible à l'adresse : `http://lipforge.ens-lyon.fr/www/crlibm/`.

[5] J. Eve. The evaluation of polynomials. *Numerische Mathematik*, (6) :17–21, 1964.

[6] C.T. Fike. Methods of evaluating polynomial approximations in function evaluation routines. *Comm. ACM*, 10(3) :175–178, 1967.

[7] C. Finot-Moreau. *Preuves et algorithmes utilisant l'arithmétique flottante normalisée IEEE*. PhD thesis, École Normle Supérieure de Lyon, 2001.

[8] S. Gal and B. Bachelis. An accurate elementary mathematical library for the IEEE floating point standard. *ACM Trans. Math. Softw.*, 17(1) :26–45, Mars 1991.

[9] J. Harrison, T. Kubaska, S. Story, and P.T.P. Tang. The computation of transcendental functions on the IA-64 architecture. *Intel Technology Journal*, 1999.

[10] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[11] C.-P. Jeannerod and G. Revy. Stability theorems for some polynomial evaluation schemes. Manuscrit, 16 juin 2006.

[12] D.E. Knuth. Evaluation of polynomials by computers. *Comm. ACM*, 5(12) :595–599, 1962.

[13] D.E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, third edition, 1987.

[14] Ch. Q. Lauter. Basic buildings blocks for a triple-double intermediate format. Rapport de recherche de l'INRIA-Rhône-Alpes (RR5702), September 2005.

[15] G. Melquiond. Gappa - génération automatique de preuves de propriétés arithmétiques. Disponible à l'adresse : `http://lipforge.ens-lyon.fr/www/gappa/`.

[16] J.-M. Muller. On the definition of ulp(x). Rapport de recherche de l'INRIA-Rhône-Alpes (RR5504), February 2005.

[17] J.-M. Muller. *Elementary Functions, Algorithms and Implementation*. Birkhauser, Boston, second edition, October 2005.

[18] M.L. Overton. *Numerical computing with IEEE floating point arithmetic*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 2001.

[19] M.S. Paterson and L.J. Stockmeyer. On the number of nonscalar multiplications necessary to evalutate polynomials. *SIAM J. Comput.*, 2 :60–66, March 1973.

[20] S. Story and P.T.P. Tang. New algorithms for improved transcendental functions on IA-64. pages 4–11, 1999.

[21] A. Ziv. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. Math. Softw.*, 17(3) :410–423, September 1991.