Différentiation automatique et formes de Taylor en analyse statique de programmes numériques

Alexandre Chapoutot et Matthieu Martel

CEA LIST, Laboratoire Modélisation et Analyse de Systèmes en Interaction
Boîte courrier 94, F91191 Gif-sur-Yvette France
{alexandre.chapoutot,matthieu.martel}@cea.fr

Résumé Des travaux récents sur l'analyse statique de programmes numériques ont montré que les techniques d'interprétation abstraite étaient adaptées à la validation de la précision des calculs en arithmétique flottante. L'utilisation des intervalles comme domaine numérique, même avec des méthodes de subdivision, induit une sur-approximation des résultats en particulier par l'existence de l'effet enveloppant (wrapping effect). Une solution utilisée pour éviter ce problème est la définition de domaines relationnels étroitement liés aux propriétés à valider. Nous allons montrer dans cet article, comment l'utilisation les techniques de différentiation automatique peuvent être utilisées pour définir des formes de Taylor permettant de définir une nouvelle analyse statique.

1 Introduction

Les règles de conception des logiciels embarqués critiques imposent, en particulier dans le domaine de l'avionique, d'apporter des garanties sur le "bon comportement" des programmes. Or, l'arithmétique à virgule flottante est de plus en plus utilisée dans ces applications, notamment à cause de la présence d'unités de calcul dédiées, ce qui nécessite la mise au point de méthodes de validation appropriées. Les techniques de test restent difficilement applicables à la détection de problèmes de précision numérique (les pires erreurs peuvent survenir pour des jeux de données très particuliers qui ne correspondent pas par exemple à des cas limites tels que les bornes du domaine d'une variable) et plusieurs travaux de recherche récents ont porté sur la validation par analyse statique de la précision de ces calculs [10,16,9]. Dans cet article, nous présentons une nouvelle technique pour l'étude de la précision numérique, adaptée à l'analyse statique par interprétation abstraite.

Plusieurs méthodes ont été proposées pour étudier la précision numérique des calculs réalisés par un programme [1,15]. Cependant, les objectifs ne sont pas, en général, la validation de comportements numériques mais la représentation des valeurs réelles en machine, c'est-à-dire l'amélioration de la précision des résultats. L'arithmétique d'intervalles permet d'encadrer une valeur réelle par deux nombres machines. Les calculs effectués avec cette arithmétique sont garantis puisqu'à chaque opération le pire cas est considéré, d'où l'apparition de l'*effet enveloppant* cause de sur-approximations. La méthode CESTAC [3] (Contrôle et Estimation STochastique des Arrondis de Calculs) consiste à exécuter plusieurs fois un programme en changeant aléatoirement la manière d'arrondir les opérations, ce qui permet d'estimer le résultat réel d'un calcul. Comme

toute méthode statistique, elle ne garantit les résultats qu'à probabilité près. Un autre exemple d'étude de la précision numérique qui a pour objectif la représentation des réels est la méthode CENA [14] (Correction des Erreurs Numériques d'Arrondi). Elle utilise des informations données par les dérivées des fonctions calculées par un programme pour compenser les erreurs d'arrondi et approcher au mieux les calculs dans les réels. En général, ces méthodes, ne sont pas bien adaptées à la validation des comportements numériques des programmes.

La théorie de l'interprétation abstraite [4] permet, dans certains cas, d'apporter formellement ces garanties. Elle a été utilisée avec succès pour prouver l'absence d'erreurs à l'exécution de programmes embarqués critiques de très grande taille [2] et s'est avérée être adaptée à l'analyse de la précision numérique comme l'ont montré les travaux [7,9,10] qui utilisent une arithmétique flottante avec erreurs permettant de calculer la distance (i.e. l'erreur) séparant le résultat flottant du résultat réel, et de tracer l'origine des erreurs. Ces informations supplémentaires sont très utiles pour la phase de correction.

Les travaux présentés dans cet article ont pour objectif d'améliorer le calcul des erreurs dans l'arithmétique flottante avec erreurs. Pour cela, nous allons utiliser la méthode de la différentiation automatique qui permet de calculer les dérivées d'une fonction et ainsi de mettre au point des approximations polynomiales de la fonction. Le couplage de cette méthode avec l'arithmétique d'intervalles nous permet d'une part de garantir les résultats des approximations et, d'autre part, de limiter l'influence de l'effet enveloppant dans le calcul des erreurs et ainsi limiter le nombre de fausses alarmes dans l'analyse de la précision numérique par interprétation abstraite. La combinaison de l'arithmétique flottante avec erreurs et de la méthode de différentiation automatique est à la base de la nouvelle arithmétique présentée dans cet article : *l'arithmétique de l'erreur différentiée*.

Le reste de cet article est organisé comme suit : La Norme IEEE 754, l'arithmétique flottante avec erreurs et la méthode de la différentiation automatique sont introduites à la section 2. La nouvelle arithmétique est définie à la section 3 ; elle a été implantée dans un prototype et des résultats expérimentaux sont présentés à la section 4.

2 Méthodes numériques utilisées

2.1 Norme IEEE 754

Dans cet article, nous supposons que l'arithmétique utilisée en machine est conforme à la norme IEEE 754 [12,6] dont les principaux points sont brièvement présentés ci-dessous. Cette norme définit l'arithmétique à virgule flottante en base 2 qui est implantée dans la majorité des processeurs actuels. Elle caractérise complètement la représentation mémoire des éléments de cette arithmétique, appelés nombres à virgule flottante, nombres flottants ou flottants. Un flottant est composé en mémoire (au niveau du bit) de trois parties : un signe s valant 1 ou -1, un exposant e compris entre e_{min} et e_{max} et une mantisse m. Le réel r associé à un nombre flottant est donné par la relation $r = s.m.2^e$.

La norme propose plusieurs types de flottants dont les plus importants sont le type simple précision et le type double précision. La précision p d'un nombre flottant corres-

pond au nombre de bits qui compose sa mantisse. De plus, pour une précision donnée les valeurs de e_{min} et e_{max} sont fixées. Par exemple, en double précision la mantisse est codée sur 53 bits, $e_{max} = 1023$ et $e_{min} = -e_{max} - 1$ ce qui correspond à un codage sur 10 bits de l'exposant.

Les opérations $+, -, \times, \div$ et $\sqrt{}$ sont définies mathématiquement par la norme ce qui rend leur implantation indépendante de la machine. La propriété partagée par ces opérations est celle de l'arrondi exact, c'est-à-dire que le résultat r d'une opération flottante o est équivalent au résultat r' obtenu par le calcul de o en précision infinie puis arrondi. La norme définit plusieurs modes d'arrondi dont les principaux sont : vers $+\infty$ qui donne un flottant plus grand que le réel, vers $-\infty$ qui donne un flottant plus petit que le réel et au plus près qui prend le flottant le plus proche du réel.

Une information importante sur les flottants est l'*ulp* (Unit in the Last Place) dont la valeur donne l'ordre de grandeur du dernier bit de la mantisse. L'ulp permet d'estimer la distances séparant deux nombres flottants, c'est-à-dire de majorer l'erreur d'arrondi. Pour un flottant f, $ulp(f) = 2^{-p+e-1}$ avec p la précision du flottant et e son exposant. La majoration de l'erreur dépend du mode d'arrondi : pour les arrondis vers les infinis l'erreur d'arrondi est majorée par $ulp(f)$ tandis qu'avec le mode au plus près la majoration est de $\frac{1}{2}ulp(f)$.

La norme définit aussi des valeurs spéciales qui ont pour but de ne pas interrompre le flot de calcul lors de l'apparition de cas particuliers. Pour cela, il existe les valeurs $+\infty$ et $-\infty$ pour matérialiser les limites de la représentation, par exemple avec l'opération $1/0$ et la valeur NaN (Not a Number) pour les valeurs non représentable, par exemple pour $\sqrt{-1}$.

2.2 Arithmétique flottante avec erreurs

Dans cette section, nous présentons la sémantique de l'erreur globale, notée $[\![.]\!]_{\mathbb{E}}$, qui est une des deux composantes de la nouvelle arithmétique introduite dans cette article. Nous présentons tout d'abord la sémantique concrète qui permet de calculer exactement la distance séparant un résultat flottant du résultat réel, puis la sémantique abstraite qui est basée sur l'arithmétique d'intervalles et qui permet, en pratique, de valider la précision numérique d'un programme.

Dans la suite de cet article, toutes les sémantiques que nous allons définir s'appuient sur le langage des expressions arithmétiques donné à la figure 1. Ce langage est composé de valeurs réelles r, de variables x, et des opérations $+, -, \times, \div$. Chaque élément d'une expression arithmétique est muni d'une étiquette unique ℓ permettant de l'identifier. Ces étiquettes représentent les points de contrôle, c'est-à-dire les points importants du programme pour l'analyse de la précision numérique.

$$a^{\ell} ::= r^{\ell} \mid x^{\ell} \mid a_0^{\ell_0} +^{\ell} a_1^{\ell_1} \mid a_0^{\ell_0} -^{\ell} a_1^{\ell_1} \mid a_0^{\ell_0} \times^{\ell} a_1^{\ell_1} \mid a_0^{\ell_0} \div^{\ell} a_1^{\ell_1}$$

FIG. 1. *Grammaire des expressions arithmétiques étiquetées.*

Nous présentons la sémantique de l'erreur globale qui permet de calculer globalement l'erreur due aux arrondis pour chaque variable du programme.

L'objectif de l'arithmétique flottante avec erreurs est de mesurer la qualité d'un calcul flottant par rapport au même calcul réalisé avec l'arithmétique réelle. Une valeur réelle r est représentée dans par un couple (f, e) avec f la valeur flottante et e l'erreur d'arrondi tel que $r = f + e$. e représente la distance séparant le flottant f du réel r. Ainsi, plus la valeur de e est petite et plus la qualité du calcul est grande. Cette arithmétique est définie suivant la structure d'une expression arithmétique a par $[\![a]\!]^\natural_{\mathbb{E}} = (\mathcal{F}^\natural(a), \mathcal{E}^\natural(a))$. La fonction $\mathcal{F}^\natural$, donnée à la figure 2(a), correspond à l'arithmétique flottante telle qu'elle est définie dans la norme IEEE 754. La fonction $\mathcal{E}^\natural$, donnée à la figure 2(b), permet de calculer l'erreur globale générée par l'expression arithmétique considérée.

Les fonctions $\mathcal{F}^\natural$ et $\mathcal{E}^\natural$ utilisent deux fonctions auxiliaires qui sont $\uparrow_\circ$: $\mathbb{R} \to \mathbb{F}$ et $\downarrow_\circ$: $\mathbb{R} \to \mathbb{R}$. La fonction $\uparrow_\circ$ associe à un réel r le nombre flottant f correspondant suivant le mode d'arrondi $\circ$ choisi. Cette fonction définit la partie représentable de r dans les flottants. La fonction $\downarrow_\circ$ calcule la partie non représentable de r dans les flottants et est définie par : $\downarrow_\circ (r) = r - \uparrow_\circ (r)$.

La fonction $\mathcal{F}^\natural$ est l'implantation directe de la norme IEEE 754. La fonction $\mathcal{E}^\natural$ permet de propager les erreurs entre les différents éléments d'une expression arithmétique. De plus, conformément à la norme IEEE 754, chaque opération introduit une nouvelle erreur issue de l'arrondi du résultat de cette opération. Par exemple, comme on peut le voir à la figure 2(b), l'erreur pour une addition est $\mathcal{E}^\natural(a+b) = e_a + e_b + \downarrow_\circ (f_a + f_b)$: les erreurs sur les deux opérandes sont additionnées et l'erreur $\downarrow_\circ (f_a + f_b)$ due à l'addition flottante $\uparrow_\circ (f_a + f_b)$ est elle-même ajoutée au résultat. La sémantique de la soustraction est similaire à celle de l'addition. Quant à celle de la multiplication, elle découle du développement de $(f_a + e_a) \times (f_b + e_b)$. La définition du calcul de l'erreur pour l'opération de l'inverse est plus compliquée puisqu'elle fait intervenir une décomposition en série entière [16,15].

$$a = (f_a, e_a) \;\; et \;\; b = (f_b, e_b)$$

$$\mathcal{F}^\natural(a+b) = \uparrow_\circ (f_a + f_b)$$
$$\mathcal{F}^\natural(a-b) = \uparrow_\circ (f_a - f_b)$$
$$\mathcal{F}^\natural(a \times b) = \uparrow_\circ (f_a \times f_b)$$
$$\mathcal{F}^\natural\left(\frac{1}{a}\right) = \uparrow_\circ \left(\frac{1}{f_a}\right)$$

(a) *Fonction* $\mathcal{F}^\natural$

$$\mathcal{E}^\natural(a+b) = e_a + e_b + \downarrow_\circ (f_a + f_b)$$
$$\mathcal{E}^\natural(a-b) = e_a - e_b + \downarrow_\circ (f_a - f_b)$$
$$\mathcal{E}^\natural(a \times b) = e_a f_b + e_b f_a + e_a e_b + \downarrow_\circ (f_a \times f_b)$$
$$\mathcal{E}^\natural\left(\frac{1}{a}\right) = \sum_{i=1}^{+\infty} (-1)^i \frac{e_a^i}{f_a^{i+1}} + \downarrow_\circ \left(\frac{1}{f_a}\right)$$

(b) *Fonction* $\mathcal{E}^\natural$

FIG. 2. *Définition des fonctions* $\mathcal{F}^\natural$ *et* $\mathcal{E}^\natural$.

Cette sémantique permet de calculer le couple flottant et erreur à chaque point de contrôle du programme. Une instabilité numérique est détectée quand la valeur de l'erreur est importante.

L'objectif de la sémantique abstraite, notée $[\![.]\!]^\sharp_{\mathbb{E}}$, est d'étudier la précision numérique d'un programme pour des classes d'exécution. Nous voulons valider le comportement numérique du programme pour des plages d'entrées possibles. Pour cela, nous abstrayons les ensembles d'entrées réelles R possibles par un couple d'intervalles $([f], [e])$. $[f]$ est une sur-approximation de l'ensemble des flottants représentant R en machine et $[e]$ est une sur-approximation de l'ensemble des erreurs $\downarrow_\circ (x),\ x \in R$. Dans la suite de cet article, tout symbole entre crochets [] dénotera un intervalle. L'arithmétique d'intervalles est plus longuement définie dans [17,1].

Nous étendons la fonction $\downarrow_\circ$ à des valeurs d'intervalles grâce à la fonction $\downarrow^I_\circ$ définie par l'équation (1). Cette fonction est paramétrée par le mode d'arrondi qui détermine l'intervalle d'erreur. Pour un intervalle $[a, b]$ avec a la borne inférieure et b la borne supérieure, l'erreur d'arrondi maximale η est donnée par la valeur de l'ulp de la plus grande borne, telle que définie à la section 2.1. L'ensemble des erreurs d'arrondi est alors donné par l'intervalle $[-\eta, \eta]$.

$$\downarrow^I_\circ ([a, b]) = ulp(\max(|a|, |b|)) \times [-1, 1] \tag{1}$$

La sémantique $[\![.]\!]^\sharp_{\mathbb{E}}$ est définie sur la structure d'une expression arithmétique a par $[\![a]\!]^\sharp_{\mathbb{E}} = (\mathcal{F}^\sharp(a), \mathcal{E}^\sharp(a))$. $\mathcal{F}^\sharp$ et $\mathcal{E}^\sharp$ sont les extensions des fonctions $\mathcal{F}^\natural$ et $\mathcal{E}^\natural$ à l'arithmétique d'intervalles.

Exemple 1 *Soit l'expression arithmétique tirée de [9]* $p = x - ax$ *avec* $x = 1$ *dans les flottants et avec une erreur comprise dans l'intervalle* $[-0.5, 0.5]$*), c'est-à-dire que* x *varie dans les réels entre* $[0.5, 1.5]$*. Nous fixons* $a = 0.65$ *et nous considérons qu'aucune erreur n'est attachée à* a*. Nous considérons, pour simplifier, que les calculs flottants n'engendrent pas de nouvelle erreur.*

$$\begin{aligned}
\mathcal{F}^\sharp(p) &= \mathcal{F}^\sharp(x) - \mathcal{F}^\sharp(ax) \\
&= [1, 1] - [0.65, 0.65] \times [1, 1] \\
&= [1, 1] \times [0.65, 0.65] \\
&= [0.65, 0.65] \\
\mathcal{E}^\sharp(p) &= \mathcal{E}^\sharp(x) - \mathcal{E}^\sharp(ax) \\
&= [-0.5, 0.5] - ([0, 0] \times [1, 1] + [-0.5, 0.5] \times [0.65, 0.65] + [0, 0] \times [-0.5, 0.5]) \\
&= [-0.5, 0.5] - [-0.325, 0.325] \\
&= [-0.825, 0.825]
\end{aligned}$$

Le résultat réel est alors dans l'intervalle $[-0.175, 1.475]$ *alors que le flottant est* 0.65 *il y a donc une erreur importante.*

L'exemple précédent montre que les dépendances entre les valeurs affectent les résultats de l'analyse. La méthode de différentiation automatique permet de combler ce manque.

2.3 Différentiation automatique

Cette partie présente la différentiation automatique [11] qui est la deuxième composante de la nouvelle arithmétique définie à la section 3.

L'idée de la différentiation automatique est de considérer un programme comme une fonction mathématique définie par composition de fonctions élémentaires. En nous restreignant aux expressions arithmétiques, les fonctions élémentaires sont les opérations $+, -, \times, \div$ et les fonctions étudiées sont toutes les compositions possibles de ces opérations. Cette méthode s'appuie sur la règle de dérivation en chaîne : si f et g sont deux fonctions différentiables alors $f \circ g$ est différentiable telle que $(f \circ g)' = f'(f \circ g')$. Cette règle donne la façon de calculer la dérivée de fonctions composées, par application des règles mathématiques usuelles de dérivation. La différentiation est réalisée suivant les variables du programme et l'on distingue deux types de variables : les *variables indépendantes* et les *variables dépendantes*. Les variables indépendantes sont la plupart du temps les variables d'entrée du programme et elles sont utilisées pour la différentiation. Les variables dépendantes sont celles pour lesquelles nous voulons calculer les dérivées et sont en général les sorties du programme. De plus, les variables sont dites *actives* si elles sont accompagnées d'une information de dérivée.

Nous définissons la sémantique $[\![.]\!]^{\natural}_{\mathbb{D}}$ associée à cette méthode suivant la structure d'une expression arithmétique a telle que $[\![a]\!]^{\natural}_{\mathbb{D}} = (\mathcal{F}^{\natural}(a), \mathcal{D}^{\natural}(a))$. $\mathcal{F}^{\natural}$ est la fonction définie à la figure 2(a) implantant la norme IEEE 754 et $\mathcal{D}^{\natural}$, définie à la figure 3, correspond au calcul des dérivées en un point. Une valeur réelle est représentée par un couple (f, d) avec f la valeur flottante et d la valeur de la dérivée au point f. Comme nous l'avons mentionné précédemment, la fonction $\mathcal{D}^{\natural}$ est définie par les règles de dérivation mathématique.

$$a = (r_a, d_a) \;\; et \;\; b = (r_b, d_b)$$

$$\begin{aligned}
\mathcal{D}^{\natural}(a + b) &= d_a + d_b \\
\mathcal{D}^{\natural}(a - b) &= d_a - d_b \\
\mathcal{D}^{\natural}(a \times b) &= d_a f_b + d_b f_a \\
\mathcal{D}^{\natural}\left(\frac{1}{a}\right) &= -\frac{d_a}{f_a^2} \;\text{ si } f_a \neq 0
\end{aligned}$$

FIG. 3. *Définition de la fonction $\mathcal{D}^{\natural}$.*

Cette méthode permet, par le biais des valeurs de la différentielle, une analyse de dépendances entre les variables présentes dans le programme. En effet, la différentielle est composée de dérivées partielles calculées par rapport aux variables actives du programmes. Si une dérivée partielle p associée à la variable x est nulle pour une variable active v alors nous pouvons conclure que v ne dépend pas de x. A l'inverse, si p a la plus grande valeur non nulle parmi les dérivées partielles composant la différentielle de v alors la variable x est la plus influente dans le calcul de v.

Exemple 2 *Reprenons l'exemple de la section 2.2 avec* $p = x - ax$ *et calculons la dérivée de* p *par rapport à* x*.* $a = (0.65, 0)$ *avec* 0.65 *la valeur flottante et* 0 *la valeur de la dérivée de* a *par rapport à* x*.* $x = (1, 1)$ *où le premier* 1 *est la valeur flottante et le second* 1 *représente la valeur de la dérivée de* x *par rapport à* x*.*

$$\begin{aligned}\mathcal{D}^\natural(p) &= \mathcal{D}^\natural(x) - \mathcal{D}^\natural(ax)\\ &= 1 - (0.65 \times 1 + 0 \times 1)\\ &= 0.35\end{aligned}$$

x *a une influence dans le calcul de* p *et nous en déduisons qu'une erreur initiale* e *sur* x *induit une approximation du résultat de* a *dans les flottants de* $0.35e$*.*

3 Arithmétique de l'erreur différentiée

Dans cette section, nous présentons une nouvelle sémantique, notée $[\![.]\!]_{\mathbb{ED}}$, basée sur les méthodes de différentiation automatique et de l'erreur globale introduites aux sections 2.2 et 2.3. Nous introduisons tout d'abord à la section 3.1 une première sémantique, notée $[\![.]\!]^1_{\mathbb{ED}}$ et permettant de calculer des approximations linéaires de l'erreur, puis nous montrerons à la section 3.2 comment cette sémantique peut être étendue pour réaliser des approximations polynomiales basées sur des polynômes de Taylor. Nous détaillerons une extension basée sur des polynômes de Taylor du second ordre, notée $[\![.]\!]^2_{\mathbb{ED}}$.

Conformément à la théorie de l'interprétation abstraite [4,5], nous allons tout d'abord définir des sémantiques concrètes pour $[\![.]\!]^1_{\mathbb{ED}}$ et $[\![.]\!]^2_{\mathbb{ED}}$, c'est-à-dire des sémantiques spécifiant exactement le comportement d'un programme au cours d'une exécution, puis nous donnerons des versions abstraites de ces sémantiques dans lesquelles les ensembles de valeurs sont abstraits par des intervalles afin de pouvoir raisonner, au cours d'une analyse statique, sur des ensembles d'exécutions d'un programme.

3.1 Différentiation au premier ordre

Sémantique concrète Nous présentons maintenant la sémantique qui permet d'approximer linéairement la fonction $\mathcal{E}^\natural$ pour chaque variable du programme. Nous partons du constat qu'à chaque point de contrôle d'une expression arithmétique (c.f. figure 1) une nouvelle erreur est introduite. L'erreur introduite au point de contrôle ℓ sera notée o^ℓ. Elle correspond soit à l'erreur de représentation d'une constante, soit à l'erreur d'arrondi du résultat d'une opération. $\mathcal{E}^\natural$ (c.f. figure 2(b)) est une fonction des variables o^ℓ dont le résultat est l'erreur globale. Nous allons alors appliquer la méthode de différentiation automatique à la fonction $\mathcal{E}^\natural$ par rapports aux variables o^ℓ. Dans la sémantique concrète $[\![.]\!]^{1\natural}_{\mathbb{ED}}$, une valeur réelle r est représentée par un triplet (f, e, d) avec f la valeur flottante, e l'erreur globale et d la différentielle de e par rapport aux variables o^ℓ. $[\![\cdot]\!]^{1\natural}_{\mathbb{ED}}$ est définie suivant la structure d'une expression arithmétique a telle que

$$[\![a]\!]^{1\natural}_{\mathbb{ED}} = (\mathcal{F}^\natural(a), \mathcal{E}^\natural(a), \mathcal{D}^{1\natural}_{\mathcal{E}}(a))$$

avec $\mathcal{F}^\natural$ l'implantation de la norme IEEE 754 (c.f. figure 2(a)) et $\mathcal{E}^\natural$ la fonction de calcul de l'erreur globale (c.f. figure 2(b)). La fonction $\mathcal{D}^{1\natural}_{\mathcal{E}}$ est la composition de la

fonction $\mathcal{E}^\natural$ et de la fonction $\mathcal{D}^\natural$ dont la définition est donnée à la figure 4, elle calcule le gradient de $\mathcal{E}^\natural$. On obtient ainsi un gradient dont les composantes sont de la forme $\partial\mathcal{E}^\natural/\partial o^\ell$.

La sémantique d'une constante réelle c au point de contrôle ℓ est donnée par le triplet $(\uparrow_\circ (c), \downarrow_\circ (c), \frac{\downarrow_\circ (c)}{\partial o^\ell})$. $\uparrow_\circ (c)$ est la partie représentable de c dans les flottants, $\downarrow_\circ (c)$ est l'erreur de représentation et $\frac{\downarrow_\circ (c)}{\partial o^\ell}$ est le vecteur dont toutes les composantes sont nulles sauf la ℓ-ième qui vaut 1. Comme pour la fonction $\mathcal{D}^\natural$, la définition de la fonction $\mathcal{D}^{1\natural}_\mathcal{E}$ s'appuie sur les règles de dérivation mathématique. Pour les opérations d'addition et de soustraction nous obtenons des combinaisons linéaires des vecteurs de dérivées partielles des opérandes auxquelles nous ajoutons le vecteur associé à la nouvelle erreur d'arrondi. Le calcul pour l'opération de multiplication se base sur la formule $ax+by+xy$ avec a et b des constantes et, x et y les variables suivant lesquelles nous différentions. Nous obtenons la différentielle $ax' + by' + x'y + xy' = x'(a+y) + y'(b+x)$ et, en posant $a = f_a$, $b = f_b$, $x = e_a$, $y = e_b$, $x' = d_a$ et $y' = d_b$, nous obtenons la définition de la figure 4.

$$a = (f_a, e_a, d_a) \;\; et \;\; b = (f_b, e_b, d_b)$$

$$\mathcal{D}^{1\natural}_\mathcal{E}(a+b) = d_a + d_b + \frac{\partial \downarrow_\circ (f_a + f_b)}{\partial o^{\ell_i}}$$

$$\mathcal{D}^{1\natural}_\mathcal{E}(a-b) = d_a - d_b + \frac{\partial \downarrow_\circ (f_a - f_b)}{\partial o^{\ell_i}}$$

$$\mathcal{D}^{1\natural}_\mathcal{E}(a\times b) = d_a(f_b + e_b) + d_b(f_a + e_b) + \frac{\partial \downarrow_\circ (f_a \times f_b)}{\partial o^{\ell_i}}$$

$$\mathcal{D}^{1\natural}_\mathcal{E}\left(\frac{1}{a}\right) = \sum_{i=1}^{+\infty}(-1)^i \frac{1}{f^{i+1}} e^{i-1} d_a + \frac{\partial \downarrow_\circ (\frac{1}{f_a})}{\partial o^{\ell_i}}$$

FIG. 4. *Définition de la fonction* $\mathcal{D}^{1\natural}_\mathcal{E}$.

Dans cette sémantique, le calcul des dérivées partielles ne permet qu'une analyse de dépendances puisque le terme d'erreur est calculé dans les réels. Par contre, nous nous servirons de leurs valeurs dans la sémantique abstraite pour calculer des ensembles d'erreurs, représentés par des intervalles, mais en réduisant de manière significative l'effet enveloppant.

Sémantique abstraite La sémantique abstraite, notée $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$, permet comme la sémantique abstraite $[\![.]\!]^\sharp_\mathbb{E}$ (c.f. figure 2.2) d'évaluer la précision numérique pour des classes d'exécutions. Nous présentons dans cette partie la manière dont nous calculons des approximations linéaires garanties de l'erreur avant de définir formellement $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$.

L'analyse par intervalles [13] introduit la notion de *fonctions d'inclusion*. A une fonction mathématique μ est associée une fonction informatique ι calculée à l'aide de l'arithmétique d'intervalles. Cette fonction ι est une fonction d'inclusion si l'image

d'un intervalle $[x]$ par μ est contenu dans l'image de $[x]$ par ι. La fonction d'inclusion naturelle est celle définie directement à partir des opérations de l'arithmétique d'intervalles. Nous notons $[f]_n$ la fonction d'inclusion naturelle associée à la fonction mathématique f. Cependant, il est possible de définir d'autres fonctions d'inclusion.

Le théorème des accroissements finis dit : "Soit f une fonction différentiable sur l'intervalle $[a, b]$, de différentielle g, alors il existe c dans $]a, b[$ tel que $f(b) = f(a) + g(c)(b - a)$". En généralisant, nous obtenons :

$$\forall x \in [a, b],\ \exists c \in]a, b[,\ f(x) = f(m) + g(c)(x - m) \tag{2}$$

où m est le milieu de l'intervalle $[a, b]$. Ces théorèmes nous affirment l'existence de cette valeur c mais pas le moyen de la calculer. En utilisant l'arithmétique d'intervalles pour calculer g nous obtenons l'ensemble des dérivées de f sur $[a, b]$ et nous obtenons la relation :

$$\forall x \in [a, b],\ f(x) \in f(m) + [g]_n([a, b])(x - m) \tag{3}$$

Par extension, nous obtenons :

$$f([a, b]) \subseteq f(m) + [g]_n([a, b])([a, b] - m) \tag{4}$$

Nous pouvons alors définir la fonction d'inclusion centrée, notée $[f]_c$ pour la fonction mathématique f, définie pour toute fonction f différentiable sur un intervalle $[x]$ dont le milieu est noté m et de différentielle g. Nous avons :

$$[f]_c([x]) = f(m) + [g]_n([x])([x] - m) \tag{5}$$

Cette définition peut être étendue sans difficulté à des fonctions manipulant des valeurs vectorielles. Grâce à cette nouvelle fonction d'inclusion nous pouvons définir un nouveau calcul d'ensemble d'erreurs en prenant comme fonction f la fonction $\mathcal{E}^\natural$ et comme intervalle $[x]$ le vecteur d'intervalles d'erreurs élémentaires, noté $\boldsymbol{O}$.

Dans la sémantique abstraite, un ensemble de valeurs réelles R est représenté par un triplet $([f], m, [d])$ avec $[f]$ l'intervalle de flottants, m le centre de l'intervalle d'erreurs et $[d]$ le vecteur de dérivées partielles. La sémantique abstraite est définie sur la structure d'une expression arithmétique a telle que $[\![a]\!]^{1\sharp}_{\mathbb{ED}} = (\mathcal{F}^\sharp(a), \mathcal{E}^\sharp_m(a), \mathcal{D}^\sharp_{\mathcal{E}}(a))$. La fonction $\mathcal{F}^\sharp$ est l'extension de la fonctions $\mathcal{F}^\natural$ à l'arithmétique d'intervalles. La fonction $\mathcal{E}^\sharp_m$, définie à la figure 5, est une adaptation de la fonction $\mathcal{E}^\sharp$ au calcul de la valeur centrée de l'erreur globale. Elle suit les mêmes règles de calcul que $\mathcal{E}^\sharp$ mais en convertissant systématiquement tous les intervalles en une valeur unique qui est leur centre. Nous notons mid la fonction, qui calcule le centre d'un intervalle, définie par $mid([a, b]) = \frac{a+b}{2}$.

La fonction $\mathcal{D}^\sharp_{\mathcal{E}}$ est une extension de la fonction $\mathcal{D}^\natural_{\mathcal{E}}$ à l'arithmétique d'intervalles où chaque occurrence d'une variable d'erreur e est remplacée par un calcul de l'approximation linéaire définie par la relation (6). Cette substitution permet de calculer l'ensemble des dérivées premières de l'intervalle d'erreur et ainsi permet de garantir les approximations. $mid(\boldsymbol{O})$ représente l'application de la fonction mid sur toutes les

$$a = ([f_a], m_a, [d_a]) \;\; et \;\; b = ([f_b], m_b, [d_b])$$

$$\begin{aligned}
\mathcal{E}^\sharp_m(a+b) &= m_a + m_b + mid(\downarrow^I_\circ ([f_a] + [f_b])) \\
\mathcal{E}^\sharp_m(a-b) &= m_a - m_b + mid(\downarrow^I_\circ ([f_a] - [f_b])) \\
\mathcal{E}^\sharp_m(a \times b) &= mid(m_a[f_b]) + mid(m_b[f_a]) + m_a m_b + mid(\downarrow^I_\circ ([f_a] \times [f_b])) \\
\mathcal{E}^\sharp_m\left(\frac{1}{a}\right) &= \sum_{i=1}^{+\infty} (-1)^i \frac{m_a^i}{[f_a]^{i+1}} + mid\left(\downarrow^I_\circ \left(\frac{1}{[f_a]}\right)\right)
\end{aligned}$$

FIG. 5. *Définition de la fonction* $\mathcal{E}^\sharp_m$.

composantes de $\boldsymbol{O}$. L'ensemble réel R est obtenu par la relation $R = [f] + E$. Ce calcul permet d'obtenir l'ensemble des dérivées de l'intervalle d'erreurs.

$$E \subseteq m + [d](\boldsymbol{O} - mid(\boldsymbol{O})) \tag{6}$$

Exemple 3 *Reprenons l'exemple de la section 2.2 avec* $p = x - ax$, $x = ([1,1], 0, [1,1]_x)$ *où* 0 *est le milieu de l'intervalle d'erreur initiale* $[-0.5, 0.5]$, *et* $[1,1]$ *est la valeur de la dérivée de* e_x *par rapport à l'erreur initiale. La valeur de* a *est* $a = ([0.65, 0.65], 0, [1,1]_a)$ *(erreur initiale nulle). Nous considérons pour simplifier que les calculs flottants n'engendrent pas de nouvelles erreurs. La notation* x_y *représente une valeur d'erreur ou de dérivée* x *associée à la variable* y.

$$\begin{aligned}
\mathcal{E}^\sharp_m(p) &= \mathcal{E}^\sharp_m(x) - (mid(\mathcal{E}^\sharp_m(a) \times \mathcal{F}^\sharp(x)) + mid(\mathcal{E}^\sharp_m(x) \times \mathcal{F}^\sharp(a)) + \mathcal{E}^\sharp_m(a) \times \mathcal{E}^\sharp_m(x)) \\
&= 0 - (mid(0 \times [1,1]) + mid(0 \times [0.65, 0.65]) + 0 \times 0 \\
&= 0 \\
\mathcal{D}^{1\sharp}_\mathcal{E}(p) &= \mathcal{D}^{1\sharp}_\mathcal{E}(x) - (\mathcal{D}^{1\sharp}_\mathcal{E}(a) \times \mathcal{F}^\sharp(x) + \mathcal{D}^{1\sharp}_\mathcal{E}(x) \times \mathcal{F}^\sharp(a) + \mathcal{D}^{1\sharp}_\mathcal{E}(x) \times \mathcal{E}^\sharp(a) + \mathcal{D}^{1\sharp}_\mathcal{E}(a) \times \mathcal{E}^\sharp(x)) \\
&= [1,1]_x - ([1,1]_a \times [1,1] + [1,1]_x \times [0.65, 0.65] + [1,1]_x \times 0 + [1,1]_a \times [-0.5, 0.5]) \\
&= [1,1]_x - ([-0.5, 0.5]_a, [0.65, 0.65]_x) \\
&= ([-0.5, 0.5]_a, [0.65, 0.65]_x)
\end{aligned}$$

Nous savons par le calcul des dérivées que p *a une erreur qui dépend de l'erreur sur* a *et de l'erreur sur* x. *Nous obtenons alors un ensemble d'erreurs avec* $\otimes$ *le produit scalaire et* $(0_a, [-0.5, 0.5]_x)$ *le vecteur d'erreurs initiales :*

$$\begin{aligned}
E &= \mathcal{E}^\sharp_m(p) + \mathcal{D}^{1\sharp}_\mathcal{E}(a) \otimes (0_a, [-0.5, 0.5]_x) \\
&= 0 + ([-0.5, 0.5]_a, [0.65, 0.65]_x) \otimes (0_a, [-0.5, 0.5]_x) \\
&= [-0.325, 0.325]
\end{aligned}$$

Nous avons alors un ensemble réel R *compris dans l'intervalle* $[0.325, 0.975]$ *avec le flottant toujours égal à* 0.65.

3.2 Différentiation d'ordre supérieur

La différentiation d'ordre supérieur permet de prendre en compte beaucoup plus d'informations de dépendance, ce qui permet de définir des approximations plus pré-

cises. Nous montrons les possibilités d'extension de la sémantique précédente par application successive de la différentiation automatique afin de calculer des dérivées d'ordre n.

La fonction $\mathcal{D}^{\natural}$ peut-être appliquée sur la fonction $\mathcal{D}_{\mathcal{E}}^{1\natural}$ pour donner la fonction $\mathcal{D}_{\mathcal{E}}^{2\natural}$ calculant les dérivées secondes de la fonction $\mathcal{E}^{\natural}$ par rapport aux o^{ℓ}. En appliquant à nouveau la fonction $\mathcal{D}^{\natural}$ sur la fonction $\mathcal{D}_{\mathcal{E}}^{2\natural}$ nous pouvons calculer les dérivées troisièmes et ainsi de suite pour calculer les dérivées n-ième.

Grâce à la connaissance des dérivées de tous ordres, nous pouvons définir une fonction d'inclusion basée sur des formes de Taylor. Le théorème des accroissements finis se généralise aux fonction différentiables n fois par la formule de Taylor-Lagrange. Et, en procédant de la même manière qu'à la section 3.1, nous pouvons définir une fonction d'inclusion de Taylor, notée $[f]_T$ telle que :

$$[f]_T([x]) = f(m) + f'(m)([x]-m) + \ldots + f^{n-1}(m)\frac{([x]-m)^{n-1}}{(n-1)!} + [f^n]_n([x])\frac{([x]-m)^n}{n!} \tag{7}$$

Cette définition s'étend aussi sans difficulté à des valeurs vectorielles. L'avantage d'une telle fonction d'inclusion est de n'utiliser l'arithmétique d'intervalle qu'à l'ordre n, en général sur des petits ensembles de valeurs, et donc le limiter grandement l'effet enveloppant.

Nous présentons une extension à l'ordre 2 de la sémantique définie à la section 3.1, notée $[\![.]\!]^2_{\mathbb{ED}}$. Une valeur réelle r dans la sémantique concrète est représentée par un quadruplet (f, e, j, h) où f et e représentent respectivement la partie représentable et non représentable de r et j et h représentent les vecteurs de dérivées partielles au premier et au second ordre de l'erreur globale suivant les o^{ℓ}. La sémantique $[\![.]\!]^{2\natural}_{\mathbb{ED}}$ est définie sur la structure d'une expression arithmétique a par $[\![a]\!]^{2\natural}_{\mathbb{ED}} = (\mathcal{F}^{\natural}(a), \mathcal{E}^{\natural}_m(a), \mathcal{D}_{\mathcal{E}}^{1\natural}(a), \mathcal{D}_{\mathcal{E}}^{2\natural}(a))$. Les fonctions $\mathcal{F}^{\natural}(a)$, $\mathcal{E}^{\natural}_m(a)$ et $\mathcal{D}_{\mathcal{E}}^{1\natural}$ sont les mêmes que précédemment et sont respectivement définies aux figures 2(a), 2(b) et 4. La fonction $\mathcal{D}_{\mathcal{E}}^{2\natural}$ est définie à la figure 6. Sa définition est obtenue de la même manière que la définition de la fonction $\mathcal{D}_{\mathcal{E}}^{1\natural}$. La difficulté vient du calcul des dérivées secondes à partir du produit de deux vecteurs de dérivées premières. Il faut réaliser une multiplication matricielle entre les deux vecteurs afin d'obtenir toutes les combinaisons linéaires possibles, d'où la transposition de vecteurs.

Comme pour la sémantique $[\![.]\!]^1_{\mathbb{ED}}$, le calcul des dérivées premières et secondes ne permettent que de faire une analyse de dépendance. Les dérivées d'ordre 2 permettent de connaître l'influence des combinaisons d'erreurs élémentaires, introduites par l'opération de multiplication en particulier, sur la valeur de l'erreur globale. Ceci, nous permet d'étudier les erreurs d'ordre supérieur, erreurs résultats du produit de deux erreurs, qui ont dans certains cas une influence non négligeable sur le calcul de l'erreur.

La sémantique abstraite, notée $[\![.]\!]^{2\sharp}_{\mathbb{ED}}$, utilise la fonction d'inclusion de Taylor d'ordre 2 pour calculer des ensembles d'erreurs. Cette fonction est donnée par l'équation 8 et elle induit un changement dans le calcul des dérivées premières définies à la section 3.1 puisque ces dérivées ne doivent plus être calculées sur la totalité de l'intervalle d'erreur mais uniquement au centre. Nous définissons à la figure 7 la fonction $\mathcal{D}_{m\mathcal{E}}^{1\sharp}$ calculant les dérivées d'ordre 1 au niveau du centre de l'intervalle d'erreur, c'est-à-dire suivant

$$a = (f_a, e_a, j_a, h_a) \ \ et \ \ b = (f_b, e_b, j_b, h_b)$$

$$\begin{aligned}
\mathcal{D}^{2\natural}_{\mathcal{E}}(a+b) &= h_a + h_b \\
\mathcal{D}^{2\natural}_{\mathcal{E}}(a-b) &= h_a - h_b \\
\mathcal{D}^{2\natural}_{\mathcal{E}}(a\times b) &= h_a f_b + h_b f_a + {}^t j_b j_a + e_a j_b + {}^t j_a j_b + e_b j_a \\
\mathcal{D}^{2\natural}_{\mathcal{E}}\left(\frac{1}{a}\right) &= \sum_{i=1}^{+\infty} (-1)^i \frac{i}{f_a^{i+1}} \left(h_a e_a^{i-1} + (i-1) j_a^2 e^{i-2} \right)
\end{aligned}$$

FIG. 6. *Définition de la fonction* $\mathcal{D}^{2\natural}_{\mathcal{E}}$.

les valeurs de la fonction $\mathcal{E}^{\sharp}_m$. Cette définition est construite de la même manière que la fonction $\mathcal{E}^{\sharp}_m$ (c.f. figure 5) et le résultat de chaque opération mettant en jeu un intervalle est réduit en son centre grâce à la fonction mid.

$$[f]_{T2}([x]) = f(m) + f'(m)([x]-m) + \frac{1}{2}([x]-m)[f'']_n([x])([x]-m) \qquad (8)$$

Un ensemble de valeurs réelles R est représenté dans la sémantique abstraite $[\![.]\!]^{2\sharp}_{\mathbb{ED}}$ par un quadruplet $([f], m, j, [h])$ avec $[f]$ l'ensemble des valeurs flottantes associées à R, m la valeur centrée de l'erreur, j le vecteur de dérivées partielles premières calculée au point m et h la matrice de dérivées secondes. Les dérivées secondes sont représentées sous forme de matrice où les lignes et les colonnes représentent les points de contrôle et où la valeur de la case d'indices i, j représente la dérivée seconde de la fonction $\mathcal{E}^{\natural}$ par rapport aux erreurs élémentaires o^{ℓ_i} et o^{ℓ_j}. Cette dérivée représente la contribution du produit de o^{ℓ_i} et de o^{ℓ_j} dans la valeur de l'erreur globale.

$$a = ([f_a], m_a, j_a, [h_a]) \ \ et \ \ b = ([f_b], m_b, j_b, [h_b])$$

$$\begin{aligned}
\mathcal{D}^{1\sharp}_{m\mathcal{E}}(a+b) &= j_a + j_b + \frac{\partial \downarrow_\circ (a+b)}{\partial o^{\ell_i}} \\
\mathcal{D}^{1\sharp}_{m\mathcal{E}}(a-b) &= j_a - j_b + \frac{\partial \downarrow_\circ (a-b)}{\partial o^{\ell_i}} \\
\mathcal{D}^{1\sharp}_{m\mathcal{E}}(a\times b) &= mid(j_a[f_b]) + mid(j_b[f_a]) + m_a j_b + m_b j_a + \frac{\partial \downarrow_\circ (a\times b)}{\partial o^{\ell_i}} \\
\mathcal{D}^{1\sharp}_{m\mathcal{E}}\left(\frac{1}{a}\right) &= \sum_{i=1}^{+\infty} (-1)^i mid\left(\frac{i}{[f_a]^{i+1}}\right) m^{i-1} j_a + \frac{\partial \downarrow_\circ (\frac{1}{a})}{\partial o^{\ell_i}}
\end{aligned}$$

FIG. 7. *Définition de la fonction* $\mathcal{D}^{1\sharp}_{m\mathcal{E}}$.

La sémantique $[\![.]\!]^{2\sharp}_{\mathbb{ED}}$ est définie sur la structure d'une expression arithmétique a telle que $[\![a]\!]^{2\sharp}_{\mathbb{ED}} = (\mathcal{F}^{\sharp}(a), \mathcal{E}^{\sharp}_{m}(a), \mathcal{D}^{1\sharp}_{m\mathcal{E}}(a), \mathcal{D}^{2\sharp}_{\mathcal{E}}(a))$. Comme pour la sémantique abstraite $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$, la fonction $\mathcal{D}^{2\sharp}_{\mathcal{E}}$ est une extension de la fonction $\mathcal{D}^{2\natural}_{\mathcal{E}}$ à l'arithmétique d'intervalle où chaque occurrence d'une variable d'erreur est remplacée par le calcul de l'approximation polynomiale de l'erreur. Cette définition nous permet de calculer l'ensemble des dérivées partielles secondes sur l'intervalle d'erreur et elle nous permet d'avoir des approximations garanties.

4 Résultats expérimentaux

Nous avons implanté la sémantique de l'erreur différentiée à l'ordre 1 et 2 dans un prototype écrit en OCaml[1] permettant d'analyser un noyau de langage impératif basé sur le langage C. Nous obtenons des résultats qui confirment l'intérêt de la composition de sémantiques dans l'étude de la précision numérique. Nous illustrons les bonnes propriétés de la sémantique $[\![.]\!]_{\mathbb{ED}}$ par deux exemples. Le premier exemple permet d'apprécier le gain de précision des résultats entre un calcul d'erreur dans les intervalles et un autre avec notre nouvelle sémantique. Cet exemple est basé sur un calcul de racine carré par une méthode de Newton. Le second exemple a été construit, à partir de la suite $x_{n+1} = x_n - ax_n$, et met en évidence l'intérêt d'une différentiation de l'erreur au second ordre.

Dans ces deux exemples, nous avons analysé les programmes en utilisant une arithmétique flottante multi-précision pour calculer les erreurs. L'arithmétique multi-précision est une implantation logicielle de l'arithmétique flottante permettant de fixer arbitrairement le nombre de bits de la mantisse. De plus, afin d'atteindre un certain niveau de précision dans les résultats des analyses, nous avons déplié les boucles avant d'appliquer l'opérateur d'élargissement, quasi indispensable en interprétation abstraite, mais difficile à définir en analyse de la précision numérique. En effet, une méthode courante pour définir cet opérateur utilise un ensemble de valeurs paliers [2]. Dans notre cas, il n'est pas possible de définir a priori des paliers servant pour l'élargissement des termes d'erreur. La technique alors adoptée est la dégradation de la précision. A chaque application de l'opérateur d'élargissement, nous diminuons le nombre de bits servant à représenter les erreurs. Cette diminution a pour effet d'augmenter la valeur de l'*ulp* et permet ainsi d'avoir un ensemble dynamique de paliers.

Le premier programme implante une méthode de Newton calculant la racine carrée d'un nombre a strictement positif ; le code source est donné à la figure 8(a). Ce calcul s'appuie sur la récurrence définie par l'équation :

$$x_{n+1} = \frac{x_n}{2}\left(3 - ax_n^2\right)$$

Le résultat du programme, c'est-à-dire la racine carrée r d'un nombre a, est $r = x_p \times a$ avec x_p la valeur du point fixe de la fonction. Nous avons analysé les propriétés numériques de ce programme avec les sémantiques $[\![.]\!]^{\sharp}_{\mathbb{E}}$ et $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$ et un dépliage initial de la boucle de 10 itérations. Notre sémantique de l'erreur différentiée permet d'améliorer

[1] http ://caml.inria.fr/

```
double xn = 0.1;
double xn1 = 0.0;
double a = [25, 25]_[-0.01, 0.01];
int cond = 0;
double temp = 0.0;
double res = 0.0;

while (cond < 1) {
  xn1 = 0.5 * xn * (3.0 - a * xn * xn);
  temp = xn1 - xn;
  if (temp < 1e-12) {
    cond = 2;
  }
  if (temp > -1e-12) {
    cond = 2;
  }
  xn = xn1;
  res = xn1 * a;
 }
```

(a) *Racine carrée par la méthode de Newton.*

```
double a = [0.8,0.9];
double b = 0.35;
double x = [0.79,0.99]_[-0.1,0.1];

while (1) {
  x = a * x * x - b * x * x;
}
```

(b) *Suite géométrique quadratique en x.*

FIG. 8. *Code source des exemples.*

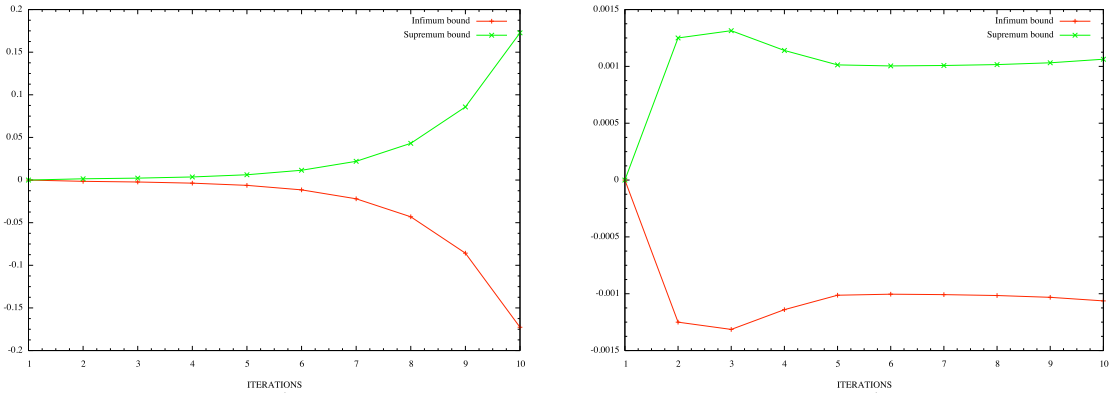
le calcul des erreurs en prenant en compte les relations qui les lient. Mais elle n'agit pas sur les valeurs flottantes. Afin d'assurer la stabilité de l'algorithme dans les flottants, nous avons choisi de calculer la racine carrée d'une valeur simple 25.0 mais entachée d'une erreur $[-0.01, 0.01]$.

L'analyse avec la sémantique $[\![.]\!]^\sharp_{\mathbb{E}}$ qui utilise l'arithmétique d'intervalles pour le calcul du flottant et de l'erreur ne permet pas d'apprécier la qualité du résultat flottant qui est $[5.0, 5.0]$ alors que l'évolution de l'intervalle d'erreur grandit. La figure 9(a) montre cette évolution en fonction des itérations. La sémantique $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$ permet, grâce à la prise en compte des relations entre erreurs, de calculer un terme d'erreur qui converge. L'évolution de l'erreur calculée par approximation linéaire est donnée à la figure 9(b). Grâce à ces informations nous pouvons conclure que le résultat flottant est très proche du résultat réel.

Le second exemple est une suite géométrique régie par l'équation $x_{n+1} = ax_n^2 - bx_n^2$. Le programme associé à cette suite est donné à la figure 8(b). Nous avons comme valeur initiale x_0 l'intervalle flottant $[0.79, 0.99]$ et celle-ci est entachée d'une erreur comprise dans l'intervalle $[-0.1, 0.1]$, c'est-à-dire que les valeurs réelles sont dans l'intervalle $[0.69, 1.09]$. La constante a est dans l'intervalle $[0.8, 0.9]$ et la constante b est égale à 0.35. Naturellement, cette suite converge positivement vers 0 quand n tend vers $+\infty$.

Une analyse par la sémantique $[\![.]\!]^\sharp_{\mathbb{E}}$ conduit à une valeur flottante qui converge vers 0 mais un terme d'erreur qui rapidement explose (à partir de la 13-ième itération) pour être de la forme $[-MAX\_FLOAT, MAX\_FLOAT]$ avec $MAX\_FLOAT = 1.7976931348623157e^{308}$. Ce résultat est uniquement dû à l'arithmétique d'intervalles qui ne permet pas de détecter que la soustraction utilise le même x_n^2.

Par contre, une analyse par la sémantique $[\![.]\!]^{1\sharp}_{\mathbb{E}}$ permet de calculer un terme d'erreur plus proche du modèle mathématique. L'évolution de l'erreur est donnée à la figure 10(a). Nous obtenons alors un terme d'erreur convergeant vers 0 au bout d'une douzaine



(a) $[\![.]\!]^{\sharp}_{\mathbb{E}}$. (b) $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$

FIG. 9. *Intervalle d'erreur pour chaque itération du programme 8(a).*

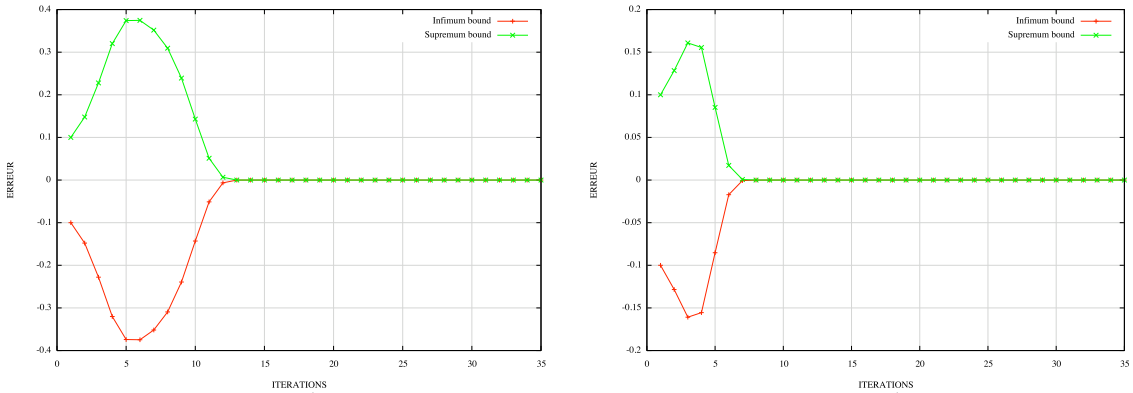
d'itérations avec une amplitude de $[-0.3742, 0.3742]$. Le passage à l'ordre 2 dans le calcul de l'erreur en utilisant la sémantique $[\![.]\!]^{2\sharp}_{\mathbb{E}}$ permet d'affiner encore ce résultat comme le montre l'évolution de l'erreur donnée à la figure 10(b). Cette sémantique montre une convergence vers 0 en 7 itérations et avec un intervalle d'erreur qui est au maximum de $[-0.161, 0.161]$.

L'élévation au carré des x_n engendre des erreurs d'ordre supérieur qui sont de plus en plus prépondérantes dans le calcul de l'erreur globale. L'application directe de l'arithmétique d'intervalles ne permet pas de détecter les relations entre les erreurs et donc fournit un mauvais résultat. La différentiation au premier ordre capte les relations entre erreurs élémentaires mais pas entre erreurs d'ordre supérieur ce qui conduit à un bon résultat mais qui peut encore être amélioré, ce que fait la sémantique à l'ordre deux.

5 Conclusion

Dans cet article, nous avons présenté une nouvelle arithmétique dédiée à l'étude de la précision numérique dans le cadre de la validation de programmes numériques. Cette arithmétique est issue de la combinaison de deux méthodes numériques existantes : l'erreur globale et la différentiation automatique. Elle permet de calculer plus finement les erreurs d'arrondi en diminuant l'effet enveloppant dans les calculs et ainsi permet d'obtenir des analyses plus précises.

Nous nous sommes concentrés sur l'amélioration du calcul des erreurs d'arrondi. Or les termes d'erreurs dépendent aussi des valeurs flottantes, en particulier dans le cas de la multiplication. Une sur-approximation de cet ensemble de flottants se répercute dans le calcul de l'erreur même avec la technique présentée ici. Une solution envisagée est d'utiliser des formes relationnelles qui introduisent des contraintes linéaires pour affiner le calcul d'ensemble de valeurs flottantes [8,9] et ainsi réduire l'effet enveloppant au niveau de la partie flottante. [9] définit une sémantique relationnelle basée sur des



(a) $[\![.]\!]^{1\sharp}_{\mathbb{ED}}$ (b) $[\![.]\!]^{2\sharp}_{\mathbb{ED}}$

FIG. 10. *Intervalle d'erreur pour chaque itération du programme 8(b).*

formes affines permettant de limiter l'effet enveloppant dans le calcul flottant mais pas dans le calcul des erreurs. Une combinaison entre cette sémantique et la sémantique de l'erreur différentiée conduirait à des analyses très précises prenant en considération les dépendances entre flottants et erreurs. [8] propose aussi de prendre en compte toutes ces dépendances, en ajoutant à la sémantique de [9] des formes affines entre les erreurs. Une comparaison de cette sémantique complètement relationnelle avec la combinaison de la sémantique de [9] et la notre est envisagée.

Par ailleurs, la connaissance des dérivées de la fonction $\mathcal{E}^{\natural}$, nous laisse entrevoir la possibilité de définir un opérateur d'élargissement adapté pour l'étude de la précision numérique. L'opérateur d'élargissement, élément important de la théorie de l'interprétation abstraite, permet d'accélérer la convergence du calcul de point fixe. Nous examinons la possibilité de définir un tel opérateur en se basant sur l'évolution des dérivées.

Références

1. J-C. Bajard, O. Beaumont, J-M. Chesneaux, M. Daumas, J. Erhel, D. Michelucci, J-M. Muller, B. Philippe, N. Revol, J-L. Roch, and J. Vignes. *Qualité des Calculs sur Ordinateurs. Vers des arithmétiques plus fiables ?* Masson, 1997.
2. B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A Static Analyzer for Large Safety-Critical Software. In *Programming Language Design and Implementation (PLDI'03)*, ACM, pages 196–207. ACM Press, 2003.
3. J.M. Chesneaux and J. Vignes. Sur la robustesse de la méthode cestac. *Comptes Rendus de l'Académie des Sciences, Paris*, 307(1) :855–860, 1988.
4. P. Cousot and R. Cousot. Abstract Interpretation : a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Principles of Programming Languages (POPL'77)*, ACM, pages 238–252. ACM Press, 1977.
5. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Symbolic Computation*, 2(4) :511–547, 1992.

6. D. Goldberg. What Every Computer Scientist Should Know About Floating-Point Arithmetic. *ACM Computing Surveys*, 23(1) :5–48, 1991.
7. E. Goubault. Static Analyses of Floatting-Point Operations. In *Static Analysis Symposium (SAS '01)*, volume 2126 of *LNCS*. Springer Verlag, 2001.
8. E. Goubault and S. Putot. Weakly relational domains for floating-point computation analysis. In *First International Workshop on Numerical and Symbolic Abstract Domains*, 2005.
9. E. Goubault and S. Putot. Static Analysis of Numerical Algorithms. In *Static Analysis Symposium (SAS '06)*, LNCS. Springer Verlag, 2006.
10. E. Goubault, S. Putot, and M. Martel. Some Future Challenges in the Validation of Control Systems. In *Embedded Real Time Software (ERTS'06)*. SIA, 2006.
11. A. Griewank. *Evaluating Derivatives : Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, 2000.
12. IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floatting-Point Arithmetic*. IEEE, New York, 1985.
13. L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.
14. P. Langlois. Automatic Linear Correction of Rounding Errors. Technical report, INRIA, 1999.
15. M. Martel. An Overview of Semantics for the Validation of Numerical Programs. In *Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*. Springer Verlag, 2005.
16. M. Martel. Semantics of roundoff error propagation in finite precision calculations. *Journal of Higher Order and Symbolic Computation*, 19 :7–30, 2006.
17. R.E. Moore. *Methods and Applications of Interval Arithmetic*. Studies in Applied Mathematics. SIAM, 1979.