Validation des résultats des logiciels scientifiques

Approche stochastique

par **Jean VIGNES**
Professeur émérite de l'université Pierre et Marie Curie
et **René ALT**
Professeur émérite de l'université Pierre et Marie Curie

Les chapitres suivants sont consacrés à l'approche stochastique de la propagation des erreurs d'arrondi et de l'influence des incertitudes des données sur les résultats fournis par un programme scientifique.

C'est la seule méthode permettant à chaque ingénieur de répondre à la question posée précédemment qui en substance est : « Quel est le nombre de chiffres décimaux significatifs exacts dans les résultats fournis par un programme de calcul scientifique ? »

Ainsi, la méthode CESTAC (Contrôle et estimation stochastique des arrondis de calculs) est détaillée au chapitre 2, puis l'arithmétique stochastique est présentée au chapitre 3.

Le chapitre 4 est consacré à la description et à l'utilisation du logiciel CADNA (« Control of Accuracy and Debugging of Numerical Algorithms »). Ce logiciel met en œuvre la méthode CESTAC et l'arithmétique stochastique discrète.

Les chapitres 5 et 6 sont dédiés à l'apport du logiciel CADNA aux diverses méthodes de calcul numérique (directes, itératives et approchées) et à des exemples d'utilisation de ce logiciel. La conclusion constitue le chapitre 7.

Toute l'introduction de ces questions est faite dans le dossier [AF 1 470], la documentation est regroupée dans [Doc. AF 1 470].

1. Estimation de l'influence des erreurs d'arrondi et des incertitudes des données

Lorsqu'un programme scientifique est exécuté sur ordinateur, les calculs sont effectués en arithmétique en virgule flottante à l'aide de données généralement entachées, soit d'erreur d'arrondi (erreurs dues à l'opérateur d'affectation), soit d'incertitudes dues aux appareils de mesures physiques (capteurs).

Les résultats fournis par l'ordinateur sont donc toujours entachés des erreurs issues de la propagation des erreurs d'arrondi et de l'influence des incertitudes des données. Il est donc toujours nécessaire d'estimer ici, et non de majorer, l'influence de ces erreurs sur tous les résultats du programme et, donc, de l'algorithme mis en œuvre.

En d'autres termes, l'utilisateur doit connaître le nombre de chiffres décimaux significatifs exacts des résultats fournis par l'ordinateur.

Soit $r \in \mathbb{R}$, le résultat exact d'un calcul numérique, et $R \in \mathbb{F}$, le résultat informatique de ce même calcul fourni par l'ordinateur, $\mathbb{F}$ étant l'ensemble des nombres représentables en virgule flottante. Intuitivement, le nombre de chiffres décimaux significatifs exacts de R est le nombre de chiffres décimaux communs à r et à R. Il est défini par :

$$C_{R,r} = \log_{10} \left| \frac{R-r}{r} \right| \quad \text{pour } r \neq 0 \tag{1}$$

Cette formule traduit le fait que, si l'erreur relative entre R et r est de l'ordre de 10^{-k}, R et r ont en commun k chiffres décimaux. Il ne faut pas se laisser abuser par l'écriture des valeurs.

En effet si, par **exemple**, $r = 4{,}239\,979$ et $R = 4{,}240\,036$, malgré le fait qu'en apparence il n'y a que deux chiffres décimaux en commun entre r et R, $C_{R,r} \simeq 5$ car la différence entre les 0 et les 9 n'est qu'illusoire.

Nous présentons dans ce chapitre une approche stochastique permettant, pour tout résultat fourni par l'ordinateur, de connaître son nombre de chiffres décimaux significatifs exacts.

1.1 Analyse des erreurs d'arrondi dues à l'arithmétique à virgule flottante

Considérons un algorithme numérique qui est une suite ordonnée de *nb* opérations arithmétiques. Pour simplifier, nous supposons qu'il ne calcule qu'un seul résultat $r \in \mathbb{R}$. Lorsque cet algorithme est mis en œuvre, à l'aide d'un langage de programmation sous forme de programme et qu'il est exécuté par l'ordinateur, le résultat fourni $R \in \mathbb{F}$ est différent de $r \in \mathbb{R}$ car il est entaché d'une erreur due à la propagation des erreurs d'arrondi de chaque opération arithmétique en virgule flottante. Toutefois, nous allons voir qu'il est possible d'estimer cette erreur.

Nous utilisons les lettres minuscules pour représenter les éléments de l'ensemble $\mathbb{R}$, et les lettres majuscules pour représenter les éléments de l'ensemble $\mathbb{F}$. Dans le même esprit, les opérateurs arithmétiques exacts sont représentés par $\{+, -, \cdot, /\}$, et les opérateurs informatiques arithmétiques en virgule flottante par $\{\oplus, \ominus, \otimes, \oslash\}$.

Tout élément $x \in \mathbb{R}$ s'exprime en virgule flottante normalisée par :

$$x = \varepsilon \cdot m \cdot b^e \quad \text{avec} \quad \frac{1}{b} \leqslant m < 1 \tag{2}$$

avec ε signe de x,

m mantisse illimitée,

b base de représentation des nombres (en général $b = 2$),

e exposant entier.

Cet élément x est représenté sur un ordinateur par un élément $X \in \mathbb{F}$ qui s'exprime par :

$$X = \varepsilon \cdot M \cdot b^E \quad \text{avec} \quad \frac{1}{b} \leqslant M < 1 \tag{3}$$

avec M mantisse de longueur p digits (p bits si $b = 2$, y compris le bit caché),

E exposant entier.

L'erreur absolue d'arrondi créée par les opérateurs informatiques est exprimée dans le cas le plus fréquent où $b = 2$ par :

– **pour l'opérateur d'affectation** :

$$X = x - \varepsilon \alpha 2^{E-p} \tag{4}$$

avec :

- pour l'arrondi au plus près : $\alpha \in [-0{,}5, 0{,}5[$,
- pour l'arrondi vers zéro : $\alpha \in [0, 1[$,
- pour l'arrondi vers $-\infty$ ou $+\infty$: $\alpha \in]-1, 1[$;

– **pour l'opérateur d'addition** $\oplus$:

$$\forall x_1, x_2 \in \mathbb{R} \text{ et } X_1, X_2 \in \mathbb{F} \text{ tels que } X_i = x_i - \varepsilon_i \alpha_i 2^{E_i - p} \tag{5}$$

$$X_1 \oplus X_2 = x_1 + x_2 - \varepsilon_1 \alpha_1 2^{E_1 - p} - \varepsilon_2 \alpha_2 2^{E_2 - p} - \varepsilon_3 \alpha_3 2^{E_3 - p} \tag{6}$$

avec E_3, ε_3 et α_3 respectivement, exposant, signe et erreur d'arrondi résultant de l'addition en virgule flottante ;

– **pour l'opérateur de soustraction** $\ominus$:

$$X_1 \ominus X_2 = x_1 - x_2 - \varepsilon_1 \alpha_1 2^{E_1 - p} + \varepsilon_2 \alpha_2 2^{E_2 - p} - \varepsilon_3 \alpha_3 2^{E_3 - p} \tag{7}$$

– **pour l'opérateur de multiplication** $\otimes$:

$$\begin{aligned} X_1 \otimes X_2 = x_1 x_2 &- \varepsilon_1 \alpha_1 x_2 2^{E_1 - p} - \varepsilon_2 \alpha_2 x_1 2^{E_2 - p} \\ &+ \varepsilon_1 \varepsilon_2 \alpha_1 \alpha_2 2^{E_1 + E_2 - 2p} - \varepsilon_3 \alpha_3 2^{E_3 - p} \end{aligned} \tag{8}$$

Dans l'équation (8) le 4e terme est du second degré en 2^{-p}. Quand ce terme est négligé, l'approximation au premier ordre de l'erreur d'arrondi due à la multiplication s'exprime par :

$$X_1 \otimes X_2 \approx x_1 x_2 - \varepsilon_1 \alpha_1 x_2 2^{E_1 - p} - \varepsilon_2 \alpha_2 x_1 2^{E_2 - p} - \varepsilon_3 \alpha_3 2^{E_3 - p} \tag{9}$$

– **pour l'opérateur de division** $\oslash$:

De la même façon que pour la multiplication, l'approximation en 1er ordre en 2^{-p} de l'erreur d'arrondi due à la division s'exprime par :

$$X_1 \oslash X_2 \approx \frac{x_1}{x_2} - \varepsilon_1 \frac{\alpha_1}{x_2} 2^{E_1 - p} + \varepsilon_2 \alpha_2 \frac{x_1}{x_2^2} 2^{E_2 - p} \tag{10}$$

1.2 Propagation des erreurs d'arrondi dans un programme de calcul scientifique

Considérons un algorithme numérique fini qui nécessite l'exécution sur ordinateur de *nb* opérations arithmétiques ordonnées et qui fournit un résultat unique $r \in \mathbb{R}$.

Après mise en œuvre de cet algorithme sur ordinateur à l'aide du programme de calcul correspondant, l'ordinateur fournira un résultat informatique entaché d'une erreur absolue due à la propagation des erreurs d'arrondi. Il a été démontré dans [25] [26] à partir des équations (4) à (10) que l'erreur absolue due à la propagation des erreurs d'arrondi, sur un résultat informatique $R \in \mathbb{F}$ nécessitant *nb* opérations arithmétiques en virgule flottante incluant l'opérateur d'affectation, s'exprime au 1er ordre en 2^{-p} par :

$$R = r + \sum_{j=1}^{j=nb} g_j\,(d)\,2^{-p}\alpha_j \qquad (11)$$

avec $g_j(d)$ quantités dépendant exclusivement des données et du programme de calcul, mais indépendantes des α_j qui sont les quantités perdues lors de l'arrondi,

r et R respectivement, résultat exact et résultat calculé par l'ordinateur.

Le nombre de bits significatifs de R est donné par :

$$C_R = -\log_2\left|\frac{R-r}{r}\right| \qquad (12)$$

Avec l'équation (11), on a :

$$C_R = -\log_2\left|\sum_{j=1}^{j=nb} g_j\,(d)\,2^{-p}\frac{\alpha_j}{r}\right| = p - \log_2\left|\sum_{j=1}^{j=nb} g_j\,(d)\,\frac{\alpha_j}{r}\right| \qquad (13)$$

Le deuxième terme de l'équation (13) représente la perte de précision dans le calcul de R. Ce terme étant indépendant de p, on peut conclure que la perte de précision, lors d'un calcul sur ordinateur, est indépendant de la précision utilisée dans le calcul. Cela veut dire que si, par exemple, en simple précision le résultat obtenu a quatre chiffres décimaux exacts (trois perdus) en double précision, le résultat aura douze chiffres décimaux exacts (trois perdus). Ceci n'est vrai que si l'approximation au 1er ordre est valable.

1.3 Influence des incertitudes des données sur les résultats d'un programme de calculs

Dans de nombreuses applications, les données sont issues d'appareils de mesures et, donc, sont entachées d'incertitudes (erreurs absolues ou relatives) chiffrées par le fabricant de ces appareils. En général, ces erreurs peuvent être considérées comme des variables aléatoires gaussiennes centrées [27]. Il est nécessaire d'estimer, d'une part, l'influence de ces incertitudes et, d'autre part, celles dues à la propagation des erreurs d'arrondi, sur tous les résultats fournis par le programme de calcul.

De la même manière qu'a été établie l'équation (11), considérons une séquence d'opérations arithmétiques fournissant un résultat unique obtenu à l'aide de données incertaines, d_i, $i = 1,..., nd$. Nous supposons que les incertitudes δ_i, $i = 1, ..., nd$ peuvent être considérées, comme des variables aléatoires gaussiennes d'écart type relatif σ_i. Il a été prouvé, dans [2] et [25] que lorsque la séquence de calcul définie ci-dessus est exécutée en arithmétique en virgule flottante en utilisant des données D_i définies par :

$$D_i = d_i\,(1 + 2\theta\sigma_i) \qquad (14)$$

avec θ nombre aléatoire distribué sur $]-1, +1[$, alors le résultat $R \in \mathbb{F}$ fourni par l'ordinateur s'exprime par :

$$R \approx r + \sum_{i=1}^{i=nd} v_i\,(d)\,2^{-p}\,\delta_i + \sum_{j=1}^{j=nb} g_j\,(d)\,2^{-p}\,\alpha_j \qquad (15)$$

avec $v_i(d)$ quantité dépendant exclusivement des données et du programme de calcul.

Cette équation est une extension de l'équation (11). En effet, la première somme représente l'effet des incertitudes des données et la seconde somme la propagation des erreurs d'arrondi.

2. Approche stochastique de l'analyse des erreurs d'arrondi : méthode CESTAC

Les méthodes présentées précédemment permettent d'estimer des bornes majorantes de la propagation des erreurs d'arrondi ou des intervalles dans lequel se trouve la solution exacte du problème étudié, et d'estimer la stabilité des algorithmes utilisés.

Depuis l'arrivée d'ordinateurs pouvant exécuter des milliards d'opérations à la seconde, les utilisateurs prennent conscience de la nécessité de connaître la fiabilité des résultats fournis par l'ordinateur après plusieurs heures de calculs arithmétiques exécutés avec l'arithmétique en virgule flottante et, souvent, avec des données entachées d'incertitudes. **Seule l'approche stochastique est capable de fournir une réponse**.

L'idée de base de l'approche stochastique est que, au cours de l'exécution d'un programme de calcul, certaines erreurs d'arrondi peuvent se compenser. Comme on ne peut pas maîtriser les erreurs d'arrondi α_i, puisqu'elles disparaissent en cours des calculs, on les considère comme des variables aléatoires uniformes équi-distribuées.

Les intervalles de variation des α_i dépendent du mode d'arrondi utilisé et sont donnés par (4). En effet, la loi de distribution de ces variables aléatoires a fait l'objet d'étude. Dans [28] et [29], il a été montré que la distribution la plus plausible pour les mantisses est une distribution logarithmique. Sous cette hypothèse, il est démontré dans [30] que les α_i au $p^{ième}$ bit pouvaient être considérées avec une très bonne approximation, comme des variables aléatoires uniformes sur leur intervalle de définition dès que $p > 10$. Or, en arithmétique en virgule flottante, $p \geqslant 24$ (voir dossier [AF 1 470]).

Avec cette approche, tout résultat $R \in \mathbb{F}$ d'une suite de calculs effectués sur ordinateur peut être considéré comme une variable aléatoire gaussienne et le nombre de chiffres significatifs exacts de ce résultat dépend des caractéristiques de cette variable aléatoire, c'est-à-dire de sa moyenne μ et de son écart-type σ. Plus le rapport $\frac{\sigma}{\mu}$ est grand et moins R a de chiffres significatifs exacts.

Mais, pour estimer μ et σ, il est nécessaire d'avoir plusieurs échantillons de la distribution de R. Malheureusement, au cours des calculs, les erreurs α_i ont disparu.

En conséquence, la question qui se pose est : « comment peut-on obtenir ces échantillons de R ? » La méthode CESTAC permet de répondre à cette question.

2.1 Base de la méthode CESTAC

La méthode CESTAC (Contrôle et estimation stochastique des arrondis de calcul) a été développée par La Porte et Vignes ([1] [31] [32], [33], [34], [35]), puis généralisée par ce dernier : [36], [37], [38], [39], [40], [41], [42].

L'idée de base de la méthode consiste à faire exécuter le même programme de calcul N fois d'une manière synchrone, c'est-à-dire que chaque opération arithmétique en virgule flottante est exécutée N fois avant d'exécuter la suivante en faisant propager différemment les erreurs d'arrondi. Ainsi, en cours de l'exécution du programme de calcul, on dispose de N échantillons de tout résultat déjà calculé. Ces N échantillons sont obtenus à l'aide du mode d'arrondi aléatoire détaillé ci-après.

2.2 Mode d'arrondi aléatoire

L'idée du mode d'arrondi aléatoire est que tout résultat d'une opération arithmétique en virgule flottante ou d'une affectation qui n'est pas une valeur flottante représentable exactement en machine est encadré par deux valeurs successives en virgule flottante, l'une par défaut (arrondie vers $-\infty$) et l'autre par excès (arrondie vers $+\infty$), chacune représentant aussi légitimement le résultat exact. L'arrondi aléatoire consiste à choisir aléatoirement l'une ou l'autre de ces valeurs avec la même probabilité 0,5.

Ainsi, lorsqu'un programme de calcul est exécuté N fois de manière synchrone en utilisant l'arrondi aléatoire, pour chaque résultat d'une opération arithmétique en virgule flottante ou pour toute affectation, N échantillons de tout résultat $R \in \mathbb{F}$ seront obtenus.

À partir de ces N échantillons, il est possible d'estimer le nombre de chiffres significatifs exacts de tout résultat.

2.3 Modélisation

Les N échantillons (R_k, $k = 1, \ldots N$) précédemment obtenus sont donc N tirages de la variable aléatoire quasi-gaussienne modélisée par l'équation (11) où les α_j sont des variables aléatoires indépendantes équi-distribuées [43]. La distribution commune des α_j est uniforme sur $[-\frac{1}{2}, +\frac{1}{2}]$ et donc centrée. Il s'agit alors d'estimer la moyenne d'une variable aléatoire gaussienne à partir de N échantillons.

C'est l'approximation de Student qui est utilisée et qui fournit, sous une probabilité donnée, l'intervalle de confiance de cette moyenne (espérance de la gaussienne). À partir de cet intervalle de confiance, on déduit le nombre de chiffres significatifs $C_{\bar{R}}$ de la moyenne $\bar{R}$ avec une probabilité β par l'équation :

$$C_{\bar{R}} = \log_{10}\left(\frac{\sqrt{n}|\bar{R}|}{\sigma \tau_\eta}\right) \qquad \textbf{(16)}$$

avec :

$$\bar{R} = \frac{1}{N}\sum_{k=1}^{N} R_k \quad \text{et} \quad \sigma^2 = \frac{1}{N-1}\sum_{k=1}^{N}(R_k - \bar{R})^2 \qquad \textbf{(17)}$$

avec τ_η valeur de la distribution de Student pour $N-1$ degrés de liberté et une probabilité de $1-\eta$.

Ainsi, la méthode CESTAC consiste à :

– faire exécuter tout programme N fois d'une manière synchrone comme expliqué précédemment, avec l'arrondi aléatoire ;

– choisir la moyenne de tout résultat calculée avec l'équation (17) comme résultat informatique ;

– calculer le nombre de chiffres décimaux de cette moyenne avec l'équation (16).

En pratique, $N = 3$ et $\eta = 0{,}05$, d'où $\tau_\eta = 4{,}303$.

2.4 Validation

Dans son utilisation pratique, la méthode CESTAC ne fournira des résultats fiables que si, et seulement si, les hypothèses que soutend le modèle théorique sont satisfaites. Les deux hypothèses fondamentales sont :

– les erreurs d'arrondi élémentaires α_j sont des variables aléatoires indépendantes uniformément distribuées et centrées (de moyenne nulle) ;

– l'approximation au 1er ordre en 2^{-p} dans la modélisation de tout résultat par l'équation (11) est valide.

En ce qui concerne la première hypothèse, l'indépendance des erreurs d'arrondi n'est pas toujours rigoureusement vérifiée, mais dans la pratique l'approximation de Student est suffisamment robuste pour donner des résultats fiables. Il arrive aussi qu'elles ne soient pas exactement centrées. Le test de Student donnera alors un estimateur biaisé du résultat exact. Mais, il est montré dans [25] et [44] qu'un biais de quelques σ entraîne une erreur inférieure à un chiffre décimal, voire à un bit, sur l'estimation de $\bar{R}$ par l'équation (16). En conséquence, même si la première hypothèse précédente n'est pas rigoureusement satisfaite, en pratique l'estimation de $C_{\bar{R}}$ par l'équation (16) n'est pas mise en défaut si l'on considère qu'elle est fournie à un chiffre décimal près.

Par contre, en ce qui concerne la seconde hypothèse, sa légitimité est fondamentale pour la validation de la méthode CESTAC.

En effet, si la seconde hypothèse n'est pas satisfaite, en pratique la moyenne $\bar{R}$ n'est plus centrée et peut être entachée d'un fort biais qui peut être prépondérant devant le résultat exact. De ce fait, l'estimation du nombre de chiffres décimaux significatifs de $\bar{R}$ par l'équation (16) n'est plus fiable.

Dans la modélisation précédente, l'approximation du deuxième ordre en 2^{-2p} n'intervient pas dans les opérations d'affectation, d'addition et de soustraction en virgule flottante, c'est-à-dire dans les équations (4), (6) et (7), mais intervient dans les multiplications et les divisions. En conséquence, seules les multiplications et divisions peuvent éventuellement mettre en défaut l'hypothèse 2.

Cependant, il est montré dans [26] et [45] que ε_1 et ε_2 étant, respectivement, les erreurs absolues d'arrondi sur les opérandes $X_1 \in \mathbb{F}$ et $X_2 \in \mathbb{F}$, si l'on a :

$$sup\left(\left|\frac{\varepsilon_1}{X_1}\right|, \left|\frac{\varepsilon_2}{X_2}\right|\right) \ll 1 \qquad \textbf{(18)}$$

Alors, l'approximation du 1er ordre en 2^{-p} est légitime. En d'autres termes, plus les opérandes sont significatifs, et plus l'approximation au 1er ordre est valide. Mais, si les opérandes deviennent non significatifs, c'est-à-dire si ε_1 et ε_2 sont de l'ordre de grandeur de X_1 et X_2, alors l'approximation au 1er ordre n'est plus valide.

En résumé, la seconde hypothèse est satisfaite en pratique si les deux conditions suivantes sont vérifiées :

– les opérandes de chaque multiplication sont tous les deux significatifs ;

– le diviseur de chaque division est significatif.

Ceci a pour conséquence pratique, lors de l'exécution d'un programme sur ordinateur, de contrôler que les conditions énoncées précédemment sont satisfaites. Si tel n'est pas le cas, cela veut

dire que l'hypothèse 2 a été violée et que les résultats fournis par l'équation (16) ne sont pas fiables. Ce contrôle est détaillé dans le chapitre suivant.

2.5 Implémentation synchrone

La nécessité de contrôler, au cours de l'exécution d'un programme sur ordinateur, les deux conditions énoncées au paragraphe 2.4, imposent de pouvoir calculer à tout moment le nombre de chiffres significatifs des résultats par l'équation (16). Pour cela, on doit disposer des N échantillons de chacun de ces résultats.

Ceci impose une implémentation synchrone de la méthode CESTAC qui consiste à faire exécuter N fois chaque opération arithmétique avec l'arrondi aléatoire avant d'exécuter la suivante. Ainsi, tout se passe comme si N programmes identiques s'exécutaient simultanément sur N ordinateurs synchronisés utilisant l'arrondi aléatoire. Pour chaque résultat, N échantillons sont obtenus permettant ainsi de calculer le nombre de chiffres décimaux significatifs de chaque résultat informatique assimilé à la moyenne des N échantillons.

Chacun de ces échantillons est obtenu à l'aide de l'arrondi aléatoire qui choisit avec une probabilité égale soit l'arrondi vers $+\infty$ ou vers $-\infty$. L'arrondi aléatoire n'opère que si le résultat de l'opération arithmétique n'est pas une valeur qui tombe juste en virgule flottante. Il ne crée donc aucune erreur artificielle. En pratique, pour éviter le cas où tous les arrondis seraient dans le même sens, les $N-1$ premiers échantillons sont créés comme décrit ci-dessus et le $N^{ième}$ est créé avec le choix opposé au choix du $N-1^{ième}$. De plus, avec l'arrondi aléatoire le théorème de l'arrondi exact est respecté.

L'implémentation synchrone de la méthode CESTAC permet donc en cours d'exécution du programme de :

– contrôler la propagation des erreurs d'arrondi au niveau de chaque opération arithmétique ;

– détecter une perte de précision pendant le calcul ;

– contrôler les débranchements ;

– détecter les violations de l'hypothèse 2 qui entraînent la non fiabilité des résultats.

L'implémentation synchrone de la méthode CESTAC permet donc d'estimer, pour tout résultat informatique, l'impact de la propagation des erreurs d'arrondi en fournissant son nombre de chiffres décimaux significatifs exacts à 1 près. Mais, elle permet aussi d'estimer l'influence des incertitudes des données sur les résultats. En effet, il suffit d'utiliser l'équation (14) pour chacune des données D_i connaissant sa valeur et son incertitude.

En conséquence, le nombre de chiffres décimaux significatifs exacts à 1 près, fourni par la machine pour tout résultat informatique aura tenu compte de la propagation des erreurs d'arrondi et des incertitudes des données.

3. Arithmétique stochastique

Du point de vue théorique, l'implémentation synchrone de la méthode CESTAC et l'utilisation de l'arrondi aléatoire transforme tout résultat informatique en une variable aléatoire quasi-gaussienne. Ainsi peut-on définir une arithmétique stochastique travaillant sur des éléments qui sont des variables aléatoires gaussiennes.

3.1 Arithmétique stochastique continue

Cette arithmétique travaille sur des opérandes appelés « nombres stochastiques », cf. [46].

Définition 1 – *nombres stochastiques*

L'ensemble des nombres stochastiques $\mathbb{S}$ est l'ensemble des variables aléatoires gaussiennes. Ainsi, $X \in \mathbb{S}$ est défini par $X = (m, \sigma)$, m étant la moyenne de X et σ sont écart-type.

Si $X \in \mathbb{S}$ et $X = (m, \sigma)$, il existe λ_η dépendant uniquement de η tel que :

$$\begin{aligned} P\,(X \in [I_{\lambda,X}]) &= 1-\eta \\ I_{\eta,X} &= [m-\lambda_\eta\,\sigma,\ m+\lambda_\eta\,\sigma] \end{aligned} \qquad \textbf{(19)}$$

avec $I_{\eta,X}$ intervalle de confiance de m pour une probabilité $1-\eta$. Pour $\eta = 0{,}05$, $\lambda_\eta = 1{,}96$. Le nombre de chiffres significatifs de m est alors obtenu par :

$$C_{\eta,X} = \log_{10}\left(\frac{|m|}{\lambda_\eta\,\sigma}\right) \qquad \textbf{(20)}$$

Définition 2 – *zéro stochastique*

$X \in \mathbb{S}$ est un zéro stochastique noté par $\underline{0}$ si et seulement si :

$$C_{\eta,X} \leqslant 0 \quad \text{ou} \quad X = (0, 0)$$

Remarquons que si $f(x) = 0$, alors $F(X) = \underline{0}$.

avec f fonction réelle quelconque,

F image informatique de f.

Définition 3 – *opérateurs stochastiques*

Les quatre opérations arithmétiques entre deux nombres stochastiques $X_1 = (m_1, \sigma_1)$ et $X_2 = (m_2, \sigma_2)$ notées ${}_s+$, ${}_s-$, ${}_s\times$, ${}_s/$ sont définies par :

$$\begin{aligned} X_1\ {}_s+\ X_2 &\overset{def}{=} (m_1+m_2,\ \sqrt{\sigma_1^2+\sigma_2^2}) \\ X_1\ {}_s-\ X_2 &\overset{def}{=} (m_1-m_2,\ \sqrt{\sigma_1^2+\sigma_2^2}) \\ X_1\ {}_s\times\ X_2 &\overset{def}{=} (m_1\times m_2,\ \sqrt{m_2^2\sigma_1^2+m_1^2\sigma_2^2}) \\ X_1\ {}_s/\ X_2 &\overset{def}{=} \left(m_1/m_2,\ \sqrt{\left(\frac{\sigma_1}{m_2}\right)^2+\left(\frac{m_1\sigma_2}{m_2}\right)^2}\right), \text{ avec } m_2 \neq 0 \end{aligned} \qquad \textbf{(21)}$$

Remarque : les deux premières formules sont exactes, les deux dernières ne sont que des approximations au premier ordre.

Définition 4 – *égalité entre deux nombres stochastiques*

X_1 est stochastiquement égal à X_2 noté : $X_1\ {}_s=\ X_2$ si et seulement si :

$$X_1\ {}_s-\ X_2 = \underline{0} \Leftrightarrow |m_1-m_2| \leqslant \lambda_\eta\sqrt{\sigma_1^2+\sigma_2^2} \qquad \textbf{(22)}$$

Définition 5 – *relations d'ordre entre deux nombres stochastiques*, soit :

X_1 est stochastiquement supérieur à X_2 noté $X_1\ {}_s>\ X_2$ si et seulement si :

$$m_1-m_2 > \lambda_\eta\sqrt{\sigma_1^2+\sigma_2^2} \qquad \textbf{(23)}$$

X_1 est stochastiquement supérieur ou égal à X_2, noté $X_1 \,{}_s\!\geqslant X_2$ si et seulement si :

$$m_1 \geqslant m_2 \text{ ou } |m_1 - m_2| \leqslant \lambda_\eta \sqrt{\sigma_1^2 + \sigma_2^2} \qquad \textbf{(24)}$$

Partant de ces définitions, les propriétés suivantes ont été démontrées dans [46] et [47] :

$$m_1 = m_2 \Rightarrow X_1 \,{}_s\!= X_2 \text{ ;}$$

$\,{}_s\!=$ est une relation réflexive et symétrique mais non transitive ;

$$X_1 \,{}_s\!> X_2 \Rightarrow m_1 > m_2 \text{ ;}$$

$$m_1 \geqslant m_2 \Rightarrow X_1 \,{}_s\!\geqslant X_2 \text{ ;}$$

$\,{}_s\!<$ est l'opposé de $\,{}_s\!>$;

$\,{}_s\!>$ est une relation transitive ;

$\,{}_s\!\geqslant$ est une relation réflexive et symétrique mais non transitive ;

$\underline{0}$ est absorbant, c'est-à-dire $\forall X \in \mathbb{S}, \underline{0} \,{}_s\!\times X_s = \underline{0}$.

Des structures algébriques de l'arithmétique stochastique ont été étudiées. Sur ce sujet, nous renvoyons le lecteur aux articles suivants : [48], [49], [50], [51] et [52].

3.2 Arithmétique stochastique discrète

Du point de vue pratique l'implémentation synchrone de la méthode CESTAC et l'utilisation de l'arrondi aléatoire transforme tout résultat informatique en un $N-$*uplet* de nombres. Ainsi, peut-on définir une arithmétique appelée arithmétique stochastique discrète qui travaille sur ces $N-$*uplets* [53].

Définition 6 – *nombres stochastiques discrets* *(Ensemble $\mathcal{S}$)*

Un nombre stochastique discret $X \in \mathcal{S}$ est le $N-$*uplet* constitué par les N échantillons de la variable aléatoire quasi-gaussienne fournis par l'implémentation synchrone de la méthode CESTAC, soit $X = (X_1, X_2, ..., X_N)$, $X_i \in \mathbb{F}$, $i = 1, 2, ..., N$.

Définition 7 – *opérateurs de l'arithmétique stochastique discrète*

Les opérandes de l'arithmétique stochastique discrète sont les nombres stochastiques discrets. Le résultat de chacune des quatre opérations numériques de l'arithmétique stochastique discrète est par définition le résultat de l'opération arithmétique correspondante fourni par la méthode CESTAC.

En d'autres termes, soient X, Y et Z des nombres stochastiques discrets et soit $\diamond$ un opérateur numérique en virgule flottante, $\diamond \in \{\oplus, \ominus, \otimes, \oslash\}$:

$$X = (X_1, ..., X_N), Y = (Y_1, ..., Y_n), Z = (Z_1, ..., Z_N)$$

Chacun des opérateurs numériques $\,{}_s\!+$, $\,{}_s\!-$, $\,{}_s\!\times$, $\,{}_s\!/$, représenté génériquement par $\,{}_s\!\diamond$, est défini par :

$$Z = X \,{}_s\!\diamond Y \Rightarrow Z\,((X_1 \diamond Y_1)^{\pm}, ..., (X_N \diamond Y_N)^{\pm}) \qquad \textbf{(25)}$$

où le symbole $\pm$ signifie que le résultat de l'opération en virgule flottante $\diamond$ a été arrondi aléatoirement vers $+\infty$ ou $-\infty$.

Ainsi chaque opérateur numérique de l'arithmétique stochastique discrète fournit un résultat qui est un $N-$*uplet* obtenu à partir de l'opérateur en virgule flottante correspondant appliqué à chaque composante des deux opérandes et arrondi aléatoirement vers $+\infty$ ou $-\infty$.

Remarque : les notations pour les opérateurs numériques de l'arithmétique stochastique discrète sont volontairement les mêmes que ceux de l'arithmétique stochastique continue.

La formule (16) permet donc d'estimer le nombre de chiffres décimaux significatifs de tout résultat obtenu à l'aide de l'arithmétique stochastique discrète.

Définition 8 – *zéro informatique* [37]

Un nombre stochastique discret $X = (X_1, X_2, ..., X_N)$ est un zéro informatique noté @.0 si l'une des deux conditions suivantes est satisfaite.

– $\forall i$, $X_i = 0$ $i = 1, ..., N$;
– $C_{\bar{X}} \leqslant 0$ **(26)**

où $C_{\bar{X}}$ est obtenu à partir de l'équation (16).

À partir du concept du zéro informatique, les relations stochastiques discrètes peuvent être définies comme l'ont été les relations stochastiques continues.

Soient X, Y, Z des nombres stochastiques discrets, les relations d'ordre discrètes sont définies comme suit :

Définition 9 – *égalité stochastique discrète*

Elle est notée $\,{}_s\!=$ et définie par :

$$X \,{}_s\!= Y \text{ si } Z = X \,{}_s\!- Y = @.0$$

Définition 10 – *inégalités stochastiques discrètes*

Elles sont notées $\,{}_s\!>$ et $\,{}_s\!\geqslant$ et sont définies par :

$$\begin{aligned} X \,{}_s\!> Y \quad &\text{si} \quad \bar{X} > \bar{Y} \quad \text{et} \quad X \,{}_s\!- Y \neq @.0 \\ X \,{}_s\!\geqslant Y \quad &\text{si} \quad \bar{X} \geqslant \bar{Y} \quad \text{ou} \quad X \,{}_s\!- Y = @.0 \end{aligned} \qquad \textbf{(27)}$$

L'arithmétique stochastique discrète est l'association de la méthode CESTAC, du concept du zéro informatique et des relations stochastiques discrètes. Cette arithmétique permet la cohérence entre l'arithmétique des ordinateurs et les instructions de tests des langages de programmation telles que : si (A ⩽ B) alors..., car elle tient compte de la précision des opérandes dans les définitions des relations d'ordre. Ainsi, l'arithmétique stochastique discrète comme l'arithmétique stochastique continue retrouve une grande partie des propriétés de l'arithmétique exacte.

Nous allons maintenant décrire le logiciel CADNA qui met en œuvre l'arithmétique stochastique discrète et la méthode CESTAC.

4. Logiciel CADNA

4.1 Introduction

Le logiciel CADNA (voir [Doc. AF 1 470]) (*Control of Accuray and Debugging for Numerical Applications*) implémente l'arithmétique stochastique discrète (ASD) présentée au chapitre 3 précédent. Compte tenu des propriétés de l'ASD, le logiciel CADNA a été construit avec trois objectifs :

– estimer pour tout résultat de programme scientifique son nombre de chiffres décimaux significatifs exacts en tenant compte, d'une part de la propagation des erreurs d'arrondi et, d'autre part de l'influence des incertitudes sur les données ;

– détecter au cours de l'exécution du programme les instabilités numériques, contrôler les débranchements et vérifier si l'hypothèse 2 qui garantit la fiabilité des résultats est satisfaite. Si tel n'est pas le cas, des messages sont inscrits à la fin de l'exécution du programme et l'utilisateur doit considérer que les résultats ne sont pas fiables et doit alors faire le débogage numérique du programme. En revanche, si aucun message n'apparaît, cela signifie que les résultats fournis sont fiables ;

– permettre, si nécessaire, le débogage numérique du programme. Cela consiste, à l'aide du débogueur symbolique du langage de programmation utilisé à trouver quelle est l'instruction qui est la cause de l'anomalie numérique repérée par CADNA. L'utilisateur peut alors essayer d'améliorer les formules numériquement

instables, par exemple, en les remplaçant par des expressions équivalentes mais plus stables numériquement.

Nous présentons ici le logiciel CADNA écrit pour le langage Fortran. La même présentation peut être faite pour les langages C et ADA. Les principales spécificités de ce logiciel sont les suivantes :

– tous les opérateurs numériques, affectation, +, –, ∗, / ont été surchargés par les opérateurs stochastiques discrets correspondants afin que lorsqu'un tel opérateur est utilisé, les opérandes et le résultat soient des nombres stochastiques discrets ;

– de la même manière tous les opérateurs de relation tels que ==, >, ⩾, <, ⩽ ont également été surchargés par leurs équivalents stochastiques de façon à ce que les propriétés de l'arithmétique stochastique discrète soient satisfaites ;

– toutes les fonctions du Fortran 77 ont également été surchargées et remplacées par leurs équivalentes en arithmétique stochastique discrète ;

– enfin, l'impression des résultats numériques a été adaptée et fournit chaque résultat avec son nombre de chiffres décimaux significatifs exacts à une unité près estimé par la formule (16) ;

– afin de pouvoir estimer les effets des incertitudes des données sur les résultats fournis par le programme de calcul, une fonction spéciale a été créée pour introduire ces incertitudes dans les données. Cette fonction doit toujours être associée à l'instruction d'affectation lorsque la donnée n'est pas une valeur représentable exactement en virgule flottante, c'est-à-dire lorsque cette valeur ne tombe pas juste en machine.

Les modifications que l'utilisateur doit faire dans le programme Fortran initial sont donc principalement de changer les déclarations de types et de modifier les instructions d'impression ou d'affichage des résultats. Dans le logiciel CADNA-Fortran, les nombres stochastiques sont des triplets ($N = 3$) ce qui veut dire que la valeur numérique de τ_η dans la formule (16) pour une fiabilité de 95 % ($\eta = 0{,}05$) est $\tau_\eta = 4{,}303$.

4.2 Mise en œuvre du logiciel

Une brève description de la mise en œuvre du logiciel est présentée ici. Pour plus de détails, consulter [45] ainsi que les programmes donnés au paragraphe 6.

La mise en œuvre initiale comporte plusieurs étapes :

– l'appel à la fonction d'initialisation de la bibliothèque ;

– la modification des types de façon à substituer aux variables de types REAL, DOUBLE PRECISION des variables de type STOCHASTIQUE ;

– une modification des entrées de façon à introduire les incertitudes ou l'arrondi sur les données ;

– une modification des sorties de façon à fournir le nombre de chiffres décimaux significatifs exacts des résultats ;

– l'utilisation des fonctions spécifiques à la bibliothèque CADNA.

En particulier le programme doit se terminer par un appel à la fonction CADNA_END qui fournit un bilan des instabilités.

4.2.1 Appel à la bibliothèque CADNA

Utiliser la bibliothèque CADNA dans un programme est, en fait, exécuter le programme avec des nombres stochastiques. Ces types stochastiques sont définis dans un module (au sens du Fortran 90) qui doivent être « visibles » par les programmes, modules et sous-programmes. Pour cela, il convient donc de faire précéder les déclarations de la pseudo-instruction :

```
USE CADNA
```

Dans le cas de programmes écrits en Fortran 77, cette pseudo-instruction doit être placée :

– avant les instructions de déclaration ;

– avant la pseudo-instruction IMPLICIT NONE ;

– après la ligne d'entrée d'un programme : PROGRAM, SUBROUTINE, FUNCTION.

Dans le cas de programmes écrits en Fortran 90, cette pseudo-instruction doit être placée :

– avant les instructions de déclaration ;

– avant la pseudo-instruction IMPLICIT NONE ;

– après la ligne d'entrée d'un programme : PROGRAM ;

– après la ligne d'entrée d'un sous-programme isolé, c'est-à-dire un sous-programme qui n'est pas interne au programme principal ou qui est contenu dans un module : SUBROUTINE ;

– après la ligne d'entrée d'une fonction isolée : FUNCTION.

4.2.2 Initialisation

Cette initialisation a pour but de permettre l'utilisation de l'arithmétique aléatoire. Elle est à ajouter à la suite des déclarations du programme principal. Elle est réalisée par l'appel d'un sous-programme qui possède quatre arguments entiers :

```
call CADNA_INIT (K1, K2, K3, K4)
```

– si K1 = – 1, le bilan d'exécution contiendra tous les messages des instabilités détectées ;

– si K1 = 0, le bilan d'exécution ne contiendra aucun message d'instabilité ;

– si K1 = *k*, le bilan d'exécution contiendra les *k* premiers messages ;

– K2 permet de connaître la nature des instabilités ;

– K3 permet de détecter les cancellations ;

– K4 permet d'initialiser le générateur aléatoire de l'arithmétique stochastique utilisé par la bibliothèque. Par défaut, la graine vaut $K_4 = 51$. Ce générateur aléatoire est indépendant du système et n'interfère pas avec d'autres générateurs aléatoires éventuellement utilisés dans le programme.

4.2.3 Modifications des types

La bibliothèque CADNA définit deux nouveaux types numériques, les types stochastiques :

```
TYPE (SINGLE_ST)  : stochastique simple précision
TYPE (DOUBLE_ST) : stochastique double précision
```

Ces types stochastiques s'utilisent avec la même syntaxe que les types numériques correspondants. Il suffit donc simplement de remplacer :

```
real                par type (single_st)
double precision    par type (double_st)
```

Dans le cas où les déclarations sont implicites (real), il suffit d'ajouter en guise de déclaration :

```
implicit type (single_st) (A – H, O – Z)
```

4.2.4 Modifications à apporter aux codes écrits en Fortran

Tous les opérateurs arithmétiques et logiques, les fonctions intrinsèques et mathématiques du Fortran dans leur écriture générique ont été redéfinis. Le code de calcul n'a donc pas à être modifié. Les arguments sont de type single_st ou double_st et les résultats des fonctions sont rendus avec les types correspondants. Les fonctions intrinsèques ont été adaptées aux nombres stochastiques, voir [45]. En ce qui concerne le calcul vectoriel uniquement, les opérateurs arithmétiques vectoriels ont aussi été redéfinis tout comme les fonctions abs, min, max, sqrt.

4.2.5 Modifications des entrées

La fonction générique READ est adaptée à la lecture des nombres en virgule flottante. Il convient de transformer ces réels lus en variables de types stochastiques.

De plus, les données ne sont généralement pas codées de manière exacte en machine. Il convient donc de tenir compte de cette erreur initiale. Par ailleurs, parfois, ces données sont entachées d'une erreur de mesure. De façon à tenir compte de l'influence des incertitudes des données sur les résultats fournis, il convient aussi de perturber initialement ces données.

Cela est réalisé par l'appel du sous-programme :

call DATA_ST (X, ERX, IER)

avec X argument de type stochastique contenant la valeur de la donnée,

ERX argument de type réel qui contient la valeur de l'erreur relative ou absolue.

Le résultat de cette fonction est :

$X_i = X_i * (1 + ERX * ALEA)$ pour i = 1 à N si IER = 0

$X_i = X_i + ERX * ALEA$ pour i = 1 à N si IER = 1

Rappelons que dans la bibliothèque CADNA, N vaut 3 :

– si ERX est omis, la perturbation sera limitée au dernier bit de la mantisse. ALEA est une variable aléatoire uniformément répartie entre – 1 et 1 ;

– si ERX est nul, aucune perturbation n'est opérée ;

– si IER est omis, il est considéré comme étant égal à 0.

Remarque très importante : si la donnée X ne tombe pas juste en machine, elle doit obligatoirement être perturbée par l'utilisation du sous-programme DATA_ST sous la forme :

call DATA_ST (X)

4.2.6 Modifications des sorties

De façon à fournir les résultats avec leur nombre de chiffres décimaux significatifs, il convient de modifier les ordres d'écriture en utilisant la fonction de CADNA :

STR (X)

Cette fonction traite un argument de type stochastique et retourne une chaîne de caractères contenant la représentation décimale avec exposant de l'argument. Seuls les chiffres décimaux significatifs exacts à une unité près sont contenus dans la chaîne. La précision est alors immédiatement lisible. Lorsque l'argument est un zéro informatique, la chaîne affichée est : @.0.

Ainsi, print*, X et write (*,*) X doivent être respectivement remplacés par print*, str(X) et write (*,*) str(X).

Remarque : le dernier chiffre significatif fourni par la fonction str n'est jamais garanti.

4.2.7 Utilisation des fonctions spécifiques de CADNA

Le logiciel CADNA fournit deux fonctions spécifiques supplémentaires :

– nb_significant_digits : traite un argument de type stochastique et renvoie une valeur de type integer représentant le nombre de chiffres décimaux significatifs exacts de cet argument ;

– old_type : traite un argument de type stochastique et renvoie la moyenne des éléments de cet argument dans une variable du type standard correspondant. On perd évidemment la connaissance du nombre de chiffres significatifs de la valeur fournie.

4.2.8 Sous-programme CADNA_END

L'instruction call cadna_end () doit obligatoirement être la dernière instruction du programme principal. Elle permet l'arrêt correct du programme et l'écriture sur la sortie standard du résultat de la détection des instabilités numériques détectées au cours de l'exécution du programme.

5. Apport du logiciel CADNA aux diverses méthodes de calcul scientifique

Les méthodes numériques, mises en œuvre dans les programmes de calcul scientifique, sont classées en trois catégories :

– finies ;

– itératives ;

– approchées.

Pour chacune d'entre elles nous allons mettre en évidence les difficultés de leur utilisation et montrer comment ces difficultés peuvent être résolues en utilisant le contrôle de la précision et les concepts de l'arithmétique stochastique discrète, c'est-à-dire en utilisant le logiciel CADNA.

5.1 Méthodes numériques finies

Ces méthodes sont constituées de suites finies et ordonnées d'opérations dépendant de débranchements conditionnels. Lorsqu'elles sont exécutées sur ordinateur, à cause de la propagation des erreurs d'arrondi, deux problèmes se posent à la fois :

– la fiabilité des résultats (nombre de chiffres décimaux significatifs exacts) ;

– la fiabilité des débranchements.

Nous illustrons ceci en présentant sur des exemples les résultats obtenus, d'une part, avec l'arithmétique en virgule flottante usuelle, utilisant la norme IEEE 754 et, d'autre part, avec le logiciel CADNA.

Exemple Le premier exemple est le calcul du déterminant de la matrice de Hilbert d'ordre n définie par :

$$H = \left(a_{ij} = \frac{1}{i+j-1} \right) \quad i, j = 1, \ldots, n$$

La valeur exacte du déterminant Δ^* est donnée par :

$$\Delta^* = \frac{v^4(n-1)}{v(2n-1)} \quad \text{avec} \quad v(n) = \prod_{i=0}^{i=n} i! \qquad \textbf{(28)}$$

Le programme de calcul est présenté dans le chapitre 6.

Afin de montrer l'apport de CADNA dans l'estimation de l'effet des incertitudes de données nous présentons dans le tableau **1** les résultats en considérant d'abord que les valeurs a_{ij} ne sont qu'entachées des erreurs d'arrondi puis qu'elles sont entachées d'incertitudes d'écart-type ε.

■ Résultats obtenus dans le **cas où les valeurs a_{ij} ne sont qu'entachées des erreurs d'arrondi**. Ils sont présentés dans le tableau **1**. Les chiffres décimaux significatifs exacts dans la première colonne sont imprimés en gras. Ils ont été obtenus par comparaison avec la solution exacte.

Comme indiqué au paragraphe 4.2.6, le logiciel CADNA n'affiche que le nombre de chiffres significatifs exacts, le dernier étant estimé à une unité près. On constate ici une concordance parfaite

Tableau 1 – Calcul du déterminant de la matrice de Hilbert

Résultats virgule flottante IEEE 754 (double précision, arrondi au plus près)	Résultats avec CADNA
n = 7	**n = 7**
pivot 1 = **0,100 000 000 000 000** 10^{1} pivot 2 = **0,833 333 333 333 333** 10^{-1} pivot 3 = **0,555 555 555 555 55**2 10^{-2} pivot 4 = **0,357 142 857 142 8**74 10^{-2} pivot 5 = **0,226 757 369 614** 673 10^{-4}	pivot 1 = 0,100 000 000 000 000 10^{1} pivot 2 = 0,833 333 333 333 333 10^{-1} pivot 3 = 0,555 555 555 555 555 10^{-2} pivot 4 = 0,357 142 857 142 8 10^{-3} pivot 5 = 0,226 757 369 61 10^{-4}
pivot 6 = **0,143 154 905 0**48 182 10^{-5} pivot 7 = **0,900 974 92**3 643 140 10^{-7} Déterminant = **0,483 580 26**0 579 847 10^{-24} Dét. exact = **0,483 580 262 392 612 1** 10^{-24}	pivot 6 = 0,143 154 905 0 10^{-5} pivot 7 = 0,900 974 92 10^{-7} Déterminant = 0,483 580 26 10^{-24}
n = 10	**n = 10**
pivot 1 = **0,100 000 000 000 000** 10^{1} pivot 2 = **0,833 333 333 333 333** 10^{-1} pivot 3 = **0,555 555 555 555 55**2 10^{-2} pivot 4 = **0,357 142 857 142 8**74 10^{-3} pivot 5 = **0,226 757 369 614** 673 10^{-4} pivot 6 = **0,143 154 905 0**48 182 10^{-5} pivot 7 = **0,900 974 92**3 643 140 10^{-7} pivot 8 = **0,565 997** 060 716 175 10^{-8} pivot 9 = **0,355 1**36 255 332 890 10^{-9} pivot 10 = **0,222 6**65 694 306 967 10^{-11} Déterminant = **0,216 4**36 779 457 214 10^{-52} Dét. exact = **0,216 417 922 643 141 7** 10^{-52}	pivot 1 = 0,100 000 000 000 000 10^{1} pivot 2 = 0,833 333 333 333 332 10^{-1} pivot 3 = 0,555 555 555 555 554 10^{-2} pivot 4 = 0,357 142 857 142 7 10^{-3} pivot 5 = 0,226 757 369 61 10^{-4} pivot 6 = 0,143 154 905 0 10^{-5} pivot 7 = 0,900 974 92 10^{-7} pivot 8 = 0,565 997 0 10^{-8} pivot 9 = 0,355 13 10^{-9} pivot 10 = 0,222 6 10^{-10} Déterminant = 0,216 4 10^{-52}

entre le nombre de chiffres estimés par la méthode CESTAC et le nombre chiffres obtenus par comparaison avec la solution exacte.

■ Les résultats obtenus **lorsque les valeurs a_{ij} sont entachées d'une incertitude d'écart-type** $\epsilon = 10^{-10}$ sont rapportés dans le tableau **2** obtenu avec CADNA.

Bien évidemment, dans le cas où les données sont imprécises, le déterminant est aussi moins précis ce qui fait qu'il n'a que deux chiffres significatifs lorsque $n = 7$ et aucun lorsque $n = 10$.

Dans cet exemple, l'effet de la propagation des erreurs d'arrondi et des incertitudes sur les données est progressif au fur et à mesure du déroulement des calculs. L'exemple suivant montre que l'effet de ces erreurs peut aussi être brutal.

Exemple. Considérons le programme et sa version avec CADNA donnés dans le tableau **3**. Le résultat exact est $r = a * b = 20{,}8005$ et les résultats obtenus avec ces programmes sont présentés dans le tableau **4**. Il est clair que le programme standard donne un résultat faux mais l'utilisateur n'en est pas prévenu.

En revanche, lors de l'affichage des résultats du programme avec CADNA il apparaît le message « *Unstable_branching* » qui signifie que lors du test la valeur sur laquelle il a lieu est non significative. De fait, les valeurs x et v sont toutes les deux significatives avec 6 chiffres décimaux et doivent être considérées comme égales au regard de la précision des calculs, leur différence doit donc être nulle et c'est la branche correspondant à l'égalité qui doit être exécutée. En d'autres termes, le zéro informatique représente ici le zéro mathématique. En ce qui concerne le programme standard, la différence calculée $d = v - x$ est négative alors qu'elle ne représente que des erreurs d'arrondi et c'est le mauvais branchement qui est exécuté.

Tableau 2 – Calcul du déterminant de la matrice de Hilbert avec coefficients imprécis

n = 7 eps = 1.10^{-10}
pivot 1 = 0,100 000 000 0 10^{1} pivot 2 = 0,833 333 333 10^{-1} pivot 3 = 0,555 555 55 10^{-2} pivot 4 = 0,357 142 8 10^{-3} pivot 5 = 0,226 75 10^{-4} pivot 6 = 0,143 1 10^{-5} pivot 7 = 0,90 10^{-7} Déterminant = 0,48 10^{-24}
n = 10 eps = 1.10^{-10}
pivot 1 = 0,100 000 000 0 10^{1} pivot 2 = 0,833 333 333 4 10^{-1} pivot 3 = 0,555 555 55 10^{-2} pivot 4 = 0,357 142 8 10^{-3} pivot 5 = 0,226 757 10^{-4} pivot 6 = 0,143 1 10^{-5} pivot 7 = 0,900 10^{-7} pivot 8 = 0,5 10^{-8} pivot 9 = @.0 pivot 10 = @.0 Déterminant = @.0

Tableau 3 – Programme avec débranchement

Programme standard	Programme avec CADNA

program debranchement
 	
real α, b, u, v, x, d, r
 	
α=141.5
b=0.147
 	
u=α**5–b**5
v=u–5.*α**4*b+10.*α**3*b**2–&
10.*a**2*b**3+5.*a*b**4
x=(α–b)**5
d=v–x
if (d>=0.)then
r=α*b
else
r=b/α
endif
print *, 'r=', r,' x=', x,' v=', v,' d=', d
 	
end program debranchement
	program debranchement
	use cadnα
	type (single_st) α, b, u, v, x, d, r
	call cadnα_init(–1)
	α=141.5
	b=0.147
	call dαtα_st (b)
	u=α**5–b**5
	v=u–5.*α**4*b+10.*α**3*b**2–&
	10.*a**2*b**3+5.*a*b**4
	x=(α–b)**5
	d=v–x
	if (d>=0.)then
	r=α*b
	else
	r=b/α
	endif
	print *, 'r=', str(r),' x=', str(x),' v=', & str(v),' d=', str(d)
	call cadnα_end ()
	end program debranchement

Tableau 4 – Résultats donnés par le programme avec débranchement

Résultats en virgule flottante IEEE (simple précision arrondi au plus près)	Résultats avec CADNA
r = 0,103 886 93 10^{-2} x= 0,564 319 64 10^{11} v = 0,564 319 60 10^{11} d = – 4 096	r = 0,208 005 0 10^{2} x = 0,564 319 10^{11} v = 0,564 319 10^{11} d = @.0

Exemple. Cet exemple a été proposé dans [54] et [55]. Il s'agit du calcul de la formule :

$$t = L - \frac{(M - N/(L-(M-N/z)/(L-(M-N/y)/z)))}{(L-(M-N/(L-(M-N/y)/z))/(L-(M-N/Z)/(L-(M-N/y)/z)))} \qquad \textbf{(29)}$$

avec :

$$L = a+b+c \qquad M = a\,(b+c)+bc \qquad N = abc$$

$$x = \frac{b+c}{2} \qquad y = \frac{b\cdot b+c\cdot c}{b+c} \qquad z = L - \frac{M-N/x}{y}$$

$$a = 3\cdot 10^{8} \qquad b = 6 \qquad c = 5$$

Cette formule a été construite pour mettre en défaut le logiciel Prosolver [56] qui implémente, de manière asynchrone et donc erronée, la méthode CESTAC.

Le résultat exact est : $t = (b^7 + c^7)/(b^6 + c^6)$, c'est-à-dire t = 358 061/62 281 = 5,749 120 919 7.

Le résultat obtenu en arithmétique IEEE 754 double précision est, quel que soit l'arrondi utilisé, $t = 3 \cdot 10^8$ et l'utilisateur n'est pas averti que ce résultat est faux. Le résultat fourni par PROSOLVER est aussi $t = 3{,}0 \cdot 10^8$. Ce logiciel est donc bien mis en défaut. Le logiciel CADNA fournit de même $t = 3{,}0 \cdot 10^8$ mais il apparaît lors de l'affichage des résultats le message « *There are 9 instabilities* » qui comme expliqué aux paragraphes 2 et 4 signifie que l'hypothèse 2 qui sous-tend la validité des résultats fournis par la méthode CESTAC a été violée. Le résultat obtenu n'est donc pas fiable et le logiciel CADNA n'est donc pas mis en défaut comme le montre la trace d'exécution de Cadna pour le calcul de cette formule reportée dans le tableau **5**.

Tableau 5 – Trace d'exécution du calcul de la formule (29) avec CADNA

CADNA software — University P. et M. Curie — LIP6 *Self-validation detection* : ON *Mathematical instabilities detection* : ON *Branching instabilities detection* : ON *Intrinsic instabilities detection* : ON *Cancellation instabilities detection* : ON
t = 0,300 000 000 000 00 10^{9}
BE CAREFUL : the self-validation detects major problem(s). *The results are NOT guaranteed* *There are 9 numerical instabilities* 5 *UNSTABLE DIVISION(S)* 0 *UNSTABLE POWER FUNCTION(S)* 0 *UNSTABLE MULTIPLICATION(S)* 0 *UNSTABLE BRANCHING(S)* 0 *UNSTABLE MATHEMATICAL FUNCTION(S)* 0 *UNSTABLE INTRINSIC FUNCTION(S)* 4 *UNSTABLE CANCELLATION(S)*

Comme il apparaît dans ces exemples, l'apport du logiciel CADNA dans les méthodes directes est donc :

– le contrôle au niveau des divisions et des multiplications de la validité de l'hypothèse 2 de la méthode CESTAC ;

– le contrôle des débranchements ;

– l'estimation du nombre de chiffres décimaux significatifs exacts de tout résultat, compte tenu de la propagation des erreurs d'arrondi et de l'influence des incertitudes sur les données.

5.2 Méthodes numériques itératives

Ces méthodes consistent, à partir d'un point initial x_0 fixé, à calculer une suite récurrente x_n, $n \in \mathbb{N}$, l'élément X_k étant calculé à partir du ou des éléments précédents. Nous considérons ici pour

simplifier les suites récurrentes définies par $x_{k+1} = \varphi(x_k)$, φ étant une application de $\mathbb{R}^m$ dans $\mathbb{R}^m$.

Du point de vue informatique, deux problèmes se posent :

– Comment arrêter le processus correctement ?

– Quelle est la précision du résultat fourni par la machine ?

Il est quasi impossible avec l'arithmétique en virgule flottante de répondre à ces deux questions. On ne peut qu'utiliser des tests d'arrêt basés sur des quantités ε choisies *a priori* avec tous les inconvénients que cela comporte. En effet, si ε est choisi trop grand, le processus est arrêté trop tôt. Si ε est choisi trop petit, le processus est arrêté trop tard. Dans ce cas, un certain nombre d'itérations inutiles ont été exécutées sans pour autant avoir amélioré la précision de la solution.

En revanche, en utilisant le logiciel CADNA, on arrête le processus itératif lorsqu'une solution informatiquement satisfaisante est atteinte, comme nous allons le montrer sur les exemples suivants.

Mais, au préalable, rappelons que les problèmes qui nécessitent pour leur résolution l'utilisation de méthodes itératives peuvent être classés en deux catégories :

– les problèmes à **solution non contrôlable** pour lesquels il n'existe pas de fonction qui s'annule à la solution du problème. Appartiennent à cette catégorie le calcul des sommes de séries, le calcul de la limite d'une suite, le calcul de la trajectoire des systèmes physiques ;

– les problèmes à **solution contrôlable** pour lesquels il existe une fonction Ψ qui s'annule à la solution du problème. Nous allons montrer sur ces deux catégories de problèmes que l'utilisation du logiciel CADNA permet de répondre aux deux questions posées précédemment en éliminant les ε arbitraires utilisés dans les tests d'arrêt classiques.

5.2.1 Problèmes à solution non contrôlable

Le test d'arrêt classique utilisé est de la forme pour $X_k \in \mathbb{F}^m$:

– si $\|X_{k+1} - X_k\| \leqslant \varepsilon$, alors stop ;

ou

– si $\|X_{k+1} - X_k\| \leqslant \varepsilon \|X_k\|$, alors stop ;

où ε est un paramètre choisi arbitrairement.

En utilisant la bibliothèque CADNA, ces tests sont remplacés par :

$$\text{Si } \|X_{k+1} - X_k\| = 0. \quad \text{alors stop}, \quad X_k \in \mathbb{S}^m$$

On décide donc d'arrêter le processus lorsque la différence entre deux itérés successifs X_{k+1} et X_k (caractérisée par la norme) n'est plus significative ce qui signifie que la transformation de X_k en X_{k+1} n'apporte plus d'information réelle.

Nous allons illustrer ceci sur deux exemples.

Exemple. Le calcul de la somme d'une série numérique :

$$S_n = \sum_{i=1}^{i=n} \frac{x^i}{i!} \qquad \textbf{(30)}$$

Il est bien connu que, quelle que soit la valeur de x, S_n converge vers e^x quand $n \to +\infty$. Le tableau **6** présente les deux programmes. Dans le programme initial, pour le test d'arrêt classique, la valeur de ε est 10^{-15} puisque l'arithmétique virgule flottante IEEE double précision, arrondie au plus près est utilisée.

Tableau 6 – Calcul de la somme d'une série

Programme initial	Programme utilisant CADNA
Program somme_serie	Program somme_serie
	use cadna
double precision s,t,α,x,s1	type(double_st) s, t, α, x,s1
	call cadna_init(–1)
x=–5.d0	x=–5.d0
do i=1,4	do i=1,4
niter=0	niter=0
x=x–5.d0	x=x–5.d0
s=1.d0	s=1.d0
t=1.d0	t=1.d0
α=1.d0	α=1.d0
do	do
niter=niter+ 1	niter=niter+1
t=t*x/α	α t=t*x/α
s1=s+t	s1=s+t
if(abs(s1–s) < 1.d–15*abs(s1)) exit	if((s1–s)== 0.) exit
s=s1	s=s1
α=α+1.d0	α=α+ 1.d0
enddo	enddo
print *,'x= ',x,' niter=',niter,' s=', s	print *,'x= ',x,' niter=',niter,' s=', str(s)
enddo	enddo
	call cadna_end ()
end Programm somme_serie	end Program somme_serie

Les résultats obtenus sont donnés dans le tableau **7**. Il en ressort qu'un même programme fournit des résultats dont la précision est très variable suivant les données :

– en virgule flottante usuelle, pour x = – 10, le résultat est fourni avec 8 chiffres décimaux significatifs, pour x = – 15, le résultat est fourni avec 5 chiffres significatifs, pour x = – 20 et x = – 25, le résultat est complètement faux mais l'utilisateur n'en est pas averti ;

– avec CADNA, les résultats sont fournis avec leur précision associée.

Exemple. Calcul de la trajectoire d'un système dynamique.

Un problème difficile est celui du calcul de la trajectoire de cycles stables dans les systèmes dynamiques chaotiques. À partir de la suite de J.M. Muller [18] définie par :

$$x_{n+1} = 111 - \frac{1\,130}{x_n} + \frac{3\,000}{x_n x_{n-1}} \qquad \textbf{(31)}$$

Nous considérons le système dynamique défini par :

$$\begin{cases} x_{n+1} = y_n \\ y_{n+1} = 111 - \dfrac{1\,130}{x_n} + \dfrac{3\,000}{x_n x_{n-1}} \end{cases} \qquad \textbf{(32)}$$

Une étude analytique montre que la trajectoire dans le plan (x, y) de ce système tend, suivant le point de départ $P_1 = (x_1, y_1)$, vers trois points fixes limites :

– si P_1 est sur l'hyperbole (h) d'équation $y = 11 - \dfrac{30}{x}$, le point limite est $P_{\ell 2} = (6, 6)$;

– si P_1 n'est pas sur l'hyperbole (h) et $P_1 \neq (5, 5)$, le point limite est $P_{\ell 1} = (100, 100)$;

– si $P_1 = (5, 5)$, le point limite est $P_{\ell 3} = (5, 5)$.

Nous nous proposons de calculer la trajectoire de ce système pour $P_1 = (2, -4)$. Ce point appartient à l'hyperbole (h). Ce calcul est effectué, d'une part, en arithmétique virgule flottante IEEE, simple précision, arrondie au plus près et, d'autre part, avec CADNA, à l'aide des programmes Fortran donnés dans le tableau **8**.

Les résultats calculés sont présentés dans le tableau **9**.

Tableau 7 – Résultats fournis par les programmes du tableau 6

	Résultats				
	Exacts	VF IEEE double précision		Avec CADNA	
x	e^x	*niter*	S_{niter}	*niter*	S_{niter}
– 10	4,539 992 976 10^{-5}	57	**4,539 992 96**2 10^{-5}	57	0,453 999 29 10^{-4}
– 15	3,059 023 205 10^{-7}	76	**3,05**8 851 615 10^{-7}	76	0,3059 10^{-6}
– 20	2,061 153 622 10^{-9}	94	5,890 015 999 10^{-10}	94	@.0
– 25	1,388 794 386 10^{-11}	106	2,847 097 557 10^{-7}	106	@.0

Tableau 8 – Programmes du système dynamique équation (32)

Programme initial	Programme avec CADNA
Program sysdyn real x(10),y(50) x(1)=2. y(1)=–4. do n=2,20 x(n)=y(n–1) y(n)=111.–1 130./y(n–1)+3 000./(y(n–1)*x(n–1)) print *, n, x(n), y(n) enddo end Program sysdyn	Program sysdyn use cadna type(single_st) x(50),y(50) call cadna_init(–1) x(1)=2. y(1)=–4. do n=2,20 x(n)=y(n–1) y(n)=111.–1130./y(n–1)+3000./(y(n–1)*x(n–1)) print *, n, str(x(n)), str(y(n)) enddo call cadna_end () end Program sysdyn

Tableau 9 – Résultats des programmes (tableau 8) du système dynamique et trace d'exécution avec Cadna

Résultats en VF simple précision			Résultats avec CADNA		
n	Xn	Yn	n	Xn	Yn
1	0,200 000 00 10^1	– 0,400 000 00 10^1	1	0,200 000 0 10^1	– 0,400 000 0 10^1
2	– 0,400 000 00 10^1	0,185 000 00 10^2	2	– 0,400 000 0 10^1	0,185 000 0$10^2$
3	0,185 000 00 10^2	0,937 837 98 10^1	3	0,185 000 0 10^2	0,937 837 10^1
4	0,937 837 98 10^1	0,780 116 46 10^1	4	0,937 837 10^1	0,780 11 10^1
5	0,780 116 46 10^1	0,715 456 01 10^1	5	0,780 11 10^1	0,715 4 10^1
6	0,715 456 01 10^1	0,680 883 03 10^1	6	0,715 4 10^1	0,681 10^1
7	0,680 883 03 10^1	0,662 275 31 10^1	7	0,681 10^1	0,7 10^1
8	0,662 275 31 10^1	0,690 497 59 10^1	8	0,7 10^1	@.0
9	0,690 497 59 10^1	0,129 524 23 10^2	9	@.0	@.0
10	0,129 524 23 10^2	0,573 011 13 10^2	10	@.0	@.0
11	0,573 011 13 10^2	0,953 217 16 10^2	11	@.0	0,10 10^3
12	0,953 217 16 10^2	0,996 946 56 10^2	12	0,10 10^3	0,100 10^3
13	0,996 946 56 10^2	0,999 810 79 10^2	13	0,100 10^3	0,100 0 10^3
14	0,999 810 79 10^2	0,999 988 40 10^2	14	0,100 0 10^3	0,100 00 10^3
15	0,999 988 40 10^2	0,999 999 24 10^2	15	0,100 00 10^3	0,100 000 0 10^3
16	0,999 999 24 10^2	0,999 999 92 10^2	16	0,100 000 0 10^3	0,100 000 0 10^3
17	0,999 999 92 10^2	0,100 000 00 10^3	17	0,100 000 0 10^3	0,100 000 0 10^3
18	0,100 000 00 10^3	0,100 000 00 10^3	18	0,100 000 0 10^3	0,100 000 0 10^3
19	0,100 000 00 10^3	0,100 000 00 10^3	19	0,100 000 0 10^3	0,100 000 0 10^3
20	0,100 000 00 10^3	0,100 000 00 10^3	20	0,100 000 0 10^3	0,100 000 0 10^3
			CADNA software – University P. et M. Curie – LIP6 BE CAREFUL : the self-validation detects major problem(s) The results are NOT guaranteed There are 9 numerical instabilities 7 UNSTABLE DIVISION(S) 0 UNSTABLE POWER FUNCTION(S) 2 UNSTABLE MULTIPLICATION(S) 0 UNSTABLE BRANCHING(S) 0 UNSTABLE MATHEMATICAL FUNCTION(S) 0 UNSTABLE INTRINSIC FUNCTION(S) 0 UNSTABLE CANCELLATION(S)		

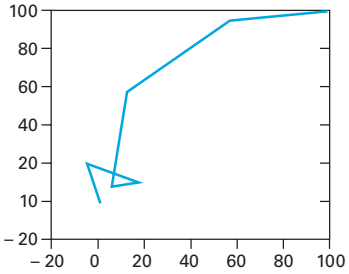

Figure 1 – Trajectoire calculée en virgule flottante

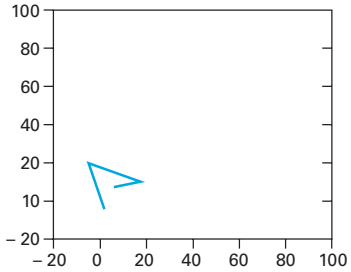

Figure 2 – Trajectoire calculée avec CADNA

Il ressort du tableau **9** et de la figure correspondante qu'en arithmétique à virgule flottante IEEE, simple précision, avec arrondi au plus près, la trajectoire converge vers le point fixe (100, 100). Ceci est en fait une fausse trajectoire, puisque, avec le point initial, la trajectoire converge mathématiquement vers (6, 6). Il est à noter que, quels que soient la précision et le mode d'arrondi de l'arithmétique virgule flottante utilisée, la trajectoire converge toujours vers (100, 100). En d'autres termes, il est impossible en virgule flottante de calculer la trajectoire de ce système.

Les résultats obtenus avec CADNA montrent qu'à la 8e itération, il y a détection d'un zéro informatique. L'affichage des résultats avec CADNA produit ici les messages « Unstable division » et « Unstable multiplication ». Cela signifie qu'à partir de la 8e itération les résultats obtenus ne sont pas fiables. Il est alors aisé avec CADNA d'arrêter le processus itératif dès que la différence entre deux itérés successifs n'est plus significative. Pour cela dans le programme précédent, il suffit d'ajouter l'instruction d'arrêt : if ||X(k) – X(k–1)|| == 0 stop. Les résultats obtenus sur cet exemple sont illustrés par les figures **1** et **2**.

La figure **1** représente la trajectoire calculée en virgule flottante IEEE 754. Cette trajectoire n'est plus significative après le 8e itéré mais l'utilisateur n'en est pas averti. La figure **2** représente la trajectoire exacte fournie avec sa précision. Le logiciel CADNA est donc un outil intéressant pour l'étude des systèmes chaotiques. Le lecteur intéressé pourra consulter les articles [57], [58] et [59].

5.2.2 Problèmes à solution contrôlable

Ces problèmes sont ceux pour lesquels il existe une fonction Ψ de $\mathbb{R}^m$ dans un autre espace $\mathbb{R}^p$ dépendant du problème et qui s'annule à la solution $x_s \in \mathbb{R}^m$, c'est-à-dire telle que $\Psi(x_s) = 0$. Par exemple :

– pour la résolution de $\mathcal{F}(\mathfrak{X}) = 0$, la fonction $\Psi(x_s)$ est $\mathcal{F}(x_s)$ elle-même ;

– pour la recherche de $\underset{Max}{Min}\ \mathcal{G}(\mathfrak{X})$, la fonction $\Psi(x_s)$ est $\Psi(x_s) = grad\ (\mathcal{G}(x_s))$.

Pour cette classe de problème, l'utilisation de CADNA qui met en œuvre, d'une part le concept du zéro informatique, et d'autre part les propriétés de l'arithmétique stochastique, il est possible d'élaborer un test d'arrêt, appelé test d'arrêt optimal basé sur la fonction Ψ qui s'annule à la solution du problème. Ce test d'arrêt optimal interrompt le processus itératif dès qu'une solution informatiquement satisfaisante est atteinte. Ce test est défini de la façon suivante :

– à chaque itération k, la valeur $\Psi(X_k)$, $X_k \in \mathcal{S}^m$ est calculée ;

– dès que $\Psi(X_k) = @.0$, le processus itératif est arrêté, car une solution informatiquement satisfaisante est atteinte. Aussi, avec l'utilisation de CADNA, le test d'arrêt optimal s'écrit :

if (Psi (x (k)) == 0.) stop

Cela suppose que le processus itératif soit convergent. Or, il est des cas où, en cours de calcul, le processus diverge. Dans ce cas le test d'arrêt optimal n'est jamais vérifié. Cela impose de toujours prévoir dans les programmes de calcul un comptage des itérations et un test d'arrêt sur le nombre maximal d'itérations fixé *a priori*. Il est aussi des cas où le processus itératif devient stationnaire. Cette stationnarité est détectée lorsque $||X_k - X_{k-1}|| = @.0$ (zéro informatique). Le processus itératif doit alors être interrompu.

En résumé, en appelant Ψ, la fonction qui doit être nulle à la solution, le processus itératif doit être interrompu à l'itération k lorsque, en cours de calcul sur ordinateur, l'une des trois éventualités suivantes se rencontre :

– $\exists k$ tel que $\Psi(X_k) = @.0$ (zéro informatique). Dans ce cas, X_k est une solution informatiquement satisfaisante ;

– $\exists k$ tel que $||X_k - X_{k-1}|| = @.0$ (zéro informatique). Dans ce cas, le processus est stationnaire. Il doit être arrêté. Il faut cependant vérifier si X_k est une solution informatiquement satisfaisante en calculant $\Psi(X_k)$;

– k est supérieur à un entier positif *Nmax* fixant le nombre maximal d'itérations. Dans ce cas, l'algorithme est non convergent.

La précision de la solution informatiquement satisfaisante X_k est fournie directement à l'impression des résultats. Pour illustrer ceci, nous présentons deux exemples.

Exemple. Résolution d'un système linéaire à coefficients réels par la méthode itérative de Jacobi. Soit à résoudre le système linéaire de dimension 20 :

$$\mathcal{A} \cdot \mathfrak{X} = \mathcal{B} \qquad \textbf{(33)}$$

Les coefficients de la matrice mathématique $\mathcal{A}$ sont générés au hasard avec la fonction RANDOM présentée au paragraphe 6. Le vecteur du second membre $\mathcal{B}$ est calculé à partir de la solution exacte $\mathfrak{X}_s$, fixée *a priori* et donc connue. Ce système a été résolu par la méthode itérative de Jacobi.

Les images informatiques de $\mathcal{A}$, $\mathcal{B}$ et $\mathfrak{X}_s$ sont notées A, B, X_s.

Pour l'utilisation du test d'arrêt classique, on fixe $\varepsilon = 3 \cdot 10^{-4}$ et la norme utilisée est $||X|| = sup\ |X_i|$ pour $i = 1,..., 20$. Les résultats obtenus sont reportés dans le tableau **10**.

Avec CADNA, le test d'arrêt optimal a été utilisé. La fonction utilisée dans le test d'arrêt est : $\Psi(X_k) = A * X_k - B$. Le processus itératif a donc été arrêté à l'itération k telle que toutes les composantes du résidu soient informatiquement nulles : $A * X_k - B = 0$. En d'autres termes, le processus itératif a été stoppé dès qu'une solution informatiquement satisfaisante a été atteinte. Ici, tous les résidus sont des zéros informatiques. De plus, la précision de la solution est fournie.

Il n'a fallu que 26 itérations pour obtenir la solution. Cet exemple illustre particulièrement l'intérêt de l'utilisation de CADNA dans les méthodes itératives. Sans que l'utilisateur ait à fournir de valeurs de ε le processus itératif est arrêté dès qu'une solution informatiquement satisfaisante est atteinte.

Tableau 10 – Résolution du système linéaire (équation 33) par la méthode de Jacobi

	Solution exacte	Résultats en VF IEEE 754 niter = 1 000		Résultats avec CADNA niter = 26	
	X_s	X_s	$\Psi(X_s)$	X_s	$\Psi(X_s)$
1	0,170 000 0 10^1	0,170 001 9 10^1	0,000 000 0 10^0	0,170 10^1	@.0
2	– 0,474 689 0 10^4	– 0,474 689 0 10^4	0,244 140 6 10^{-2}	– 0,474 689 10^4	@.0
3	0,502 300 0 10^2	0,502 300 7 10^2	0,000 000 0 10^0	0,502 3 10^2	@.0
4	– 0,245 320 1 10^3	– 0,245 320 4 10^3	0,976 562 5 10^{-3}	– 0,245 31 10^3	@.0
5	0,477 829 0 10^4	0,477 829 0 10^4	– 0,195 312 5 10^{-2}	0,477 829 10^4	@.0
6	– 0,757 300 0 10^2	– 0,757 300 1 10^2	0,000 000 0 10^0	– 0,757 3 10^2	@.0
7	0,349 543 0 10^4	0,349 543 0 10^4	0,268 554 7 10^{-2}	0,349 543 10^4	@.0
8	0,435 000 0 10^1	0,435 079 1 10^1	– 0,244 140 6 10^{-3}	0,435 10^1	@.0
9	0,452 980 0 10^3	0,452 980 2 10^3	0,000 000 0 10^0	0,452 97 10^3	@.0
10	– 0,276 000 0 10^1	– 0,275 952 7 10^1	0,000 000 0 10^0	– 0,275 10^1	@.0
11	0,823 924 0 10^4	0,823 924 0 10^4	0,341 796 9 10^{-2}	0,823 924 10^4	@.0
12	0,346 000 0 10^1	0,346 003 9 10^1	0,000 000 0 10^0	0,346 0 10^1	@.0
13	0,100 000 0 10^4	0,100 000 0 10^4	0,976 562 5 10^{-3}	0,999 99 10^3	@.0
14	– 0,500 000 0 10^1	– 0,500 084 3 10^1	0,000 000 0 10^0	– 0,500 10^1	@.0
15	0,364 240 0 10^4	0,364 240 0 10^4	0,195 312 5 10^{-2}	0,364 240 0 10^4	@.0
16	0,735 360 0 10^3	0,735 360 2 10^3	– 0,976 562 5 10^{-3}	0,735 36 10^3	@.0
17	0,170 000 0 10^1	0,169 976 3 10^1	– 0,292 968 8 10^{-2}	0,169 10^1	@.0
18	– 0,234 917 0 10^4	– 0,234 917 0 10^4	0,000 000 0 10^0	– 0,234 917 10^4	@.0
19	– 0,824 752 0 10^4	– 0,824 752 0 10^4	– 0,976 562 5 10^{-3}	– 0,824 751 10^4	@.0
20	0,984 357 0 10^4	0,984 356 9 10^4	0,781 250 0 10^{-2}	0,984 357 7 10^4	@.0

Exemple. Résolution d'une équation de type $f(x) = 0$ par la méthode de Newton.

Nous considérons l'équation polynômiale de degré 4 définie par :

$$f(x) = 0{,}1\,x^4 - 100{,}2\,x^3 + 25\,200{,}1\,x^2 - 50\,100\,x + 25\,000 \qquad x \in \mathbb{R}$$

qui admet comme racines :

$$x_1^* = x_2^* = x_3^* = x_4^* = 500$$

Nous prenons comme point de départ : $x_0 = 7\,000$.

Pour ce polynôme de degré seulement 4, la formulation de Horner n'a pas été utilisée. Comme dans l'exemple précédent, les images informatiques des variables et des fonctions réelles sont notées en lettres capitales, par exemple $X \in \mathbb{F}$ est l'image de $x \in \mathbb{R}$.

Les deux programmes sont présentés dans le tableau **11**.

Les résultats obtenus en virgule flottante IEEE, simple précision arrondie au plus près sont présentés dans le tableau **12**. Il ressort du tableau **12** que :

– il n'est pas possible d'obtenir la solution avec plus de trois chiffres décimaux significatifs ;

– lorsque ε est choisi trop petit, un grand nombre d'itérations inutiles sont effectuées sans pour autant améliorer la précision de la solution ;

– lorsque ε est choisi trop grand, le processus itératif est arrêté trop tôt. La solution fournie n'est pas la meilleure que peut fournir la machine.

En revanche, les résultats obtenus avec CADNA en simple précision sont les suivants :

$$\text{niter} = 21 \quad X = 0{,}500\ 10^3 \quad F = @.0 \quad FP = @.0$$

Ces résultats montrent que :

– le test d'arrêt optimal a interrompu le processus itératif dès qu'une solution informatiquement satisfaisante a été atteinte : $F(X_k) = @.0$;

– la solution est fournie avec son nombre de chiffres décimaux significatifs exacts : ici, trois chiffres ;

– la dérivée $FP(X_k)$ est aussi un zéro informatique ce qui laisse supposer que la racine ainsi déterminée est multiple.

L'intérêt majeur de l'utilisation du logiciel CADNA, dans les programmes de calculs scientifiques mettant en œuvre des méthodes itératives, est que, d'une part, le processus itératif est interrompu dès qu'une solution informatiquement satisfaisante est atteinte, ce qui permet d'optimiser le programme, et que, d'autre part, la précision de la solution est fournie.

De plus, avec CADNA le paramètre ε utilisé dans les tests d'arrêt classiques dont la valeur est très difficile à choisir est éliminé.

5.3 Méthodes numériques approchées

Du point de vue mathématique, ces méthodes ne fournissent qu'une approximation de la solution exacte. La solution fournie est donc toujours entachée d'une erreur de méthode e_m. Appartiennent à cette catégorie toutes les méthodes de type différences finies ou éléments finis. Du point de vue informatique, lorsque de telles méthodes sont mises en œuvre sur ordinateur, elles fournissent toujours une solution entachée d'une erreur globale e_g, résultat de la composition de l'erreur de méthode e_m inhérente à la méthode utilisée et de l'erreur due à la propagation des erreurs d'arrondi, appelée ici erreur de calcul e_c.

Il est bien connu que, lorsque le pas h de discrétisation de ces méthodes décroît, e_m décroît et qu'au contraire, quand h croît e_m croît aussi. Il a été montré [60] que lorsque h croît e_c décroît, et que, lorsque h décroît, e_c croît, de telle sorte que e_m et e_c jouent en sens opposé et que, de ce fait, l'erreur globale e_g est une fonction qui présente un minimum en fonction de h. Ainsi la meilleure approximation de la solution que l'on peut obtenir sur ordinateur correspond à un pas de discrétisation optimal h^* tel que $\dfrac{de_g}{dh} = 0$.

Tableau 11 – Programmes de résolution de l'équation polynômiale

Programme initial	Programme avec CADNA
program solpol4	program solpol4
	use cadna
real x,f,fp,y,a,b,c,d,e,eps	type(single_st) x,f,fp,y,a,b,c,d,e
	call cadna_init(–1)
print*,'entrez x0'	print*,'entrez x0'
read*,x0	read*,x0
print*,'entrez eps'	
read*,eps	
a=0.1	a=0.1
b=–100.2d0	b=–100.2d0
c=25200.1d0	c=25200.1d0
d=–50100.d0	d=–50100.d0
e=25000.d0	e=25000.d0
	call data_st(a)
	call data_st(b)
	call data_st(c)
y=x0	y=x0
x=x0+1000.	f=1.
niter=0	niter=0
do while(abs(y–x)>eps*abs(y).and.niter<500)	do while(f/=0..and.niter<500)
niter=niter+1	niter=niter+1
x=y	x=y
f=a*x**4+b*x**3+c*x**2+d*x+e	f=a*x**4+b*x**3+c*x**2+d*x+e
fp=4*a*x**3+3.*b*x**2+2.*c*x+d	fp=4*a*x**3+3.*b*x**2+2.*c*x+d
if(fp==0.) then	if(fp==0.) then
print* 'dérivée nulle'	print* 'dérivée nulle'
print* y,f,fp	print* str(y),str(f),str(fp)
stop	stop
endif	endif
y=y–f/fp	y=y–f/fp
enddo	enddo
print*, 'niter=',niter,'x=',y	print*, 'niter=',niter,'x=',str(y)
print*, f=',f,'fp=',fp	print*, f=',str(f),'fp=',str(fp)
	call cadna_end ( )
end program solpol4	end program solpol4

Tableau 12 – Solutions données par le programme initial du tableau 11

ϵ	niter	x	f	fp
10^{-5}	500	499,954 28	559,173 7	4 261,74
10^{-4}	500	499,954 28	559,173 7	4 261,74
10^{-3}	20	500,333 4	12 190,90	34 397,26
10^{-2}	17	502,755 7	767 886,2	280 684,9

En supposant que e_m et e_c ont des évolutions du même ordre de grandeur en fonction du pas h, cet optimum est obtenu lorsque $e_m \approx e_c$.

Comme e_m est connue par l'analyse mathématique de la méthode et que e_c peut être estimée avec le logiciel CADNA, il est possible de déterminer h^* et ainsi d'obtenir la meilleure solution informatique possible.

Pour illustrer ceci, considérons l'intégration numérique par la méthode d'Euler du problème de Cauchy muni de conditions initiales et défini par l'équation (34) :

$$\begin{cases} y' = e^x y + xy - (x+1)e^{-x} - 1 \\ x_0 = 0, \quad y_0 = 1 \end{cases} \qquad \textbf{(34)}$$

et qui admet comme solution exacte $y^* = e^{-x}$.

Comme nous l'avons vu ci-dessus, à chaque étape numérotée k de l'intégration, il existe un pas optimal appelé « pas optimal local » tel que dans l'intervalle $[x_k, x_{k+1}]$ on ait : $e_m \approx e_c$. L'estimation de ce pas optimal local h_k^* nécessite trois étapes :

– l'estimation de l'erreur de calcul e_c ;

– l'évaluation de l'erreur de méthode e_m ;

– le calcul du pas optimal local h_k^*. Voir le programme du tableau **19**.

Tableau 13 – Solution de l'équation (34) avec différents pas d'intégration

x	sol. exacte. y^*	solution calculée y			
		$h = 10^{-1}$	$h = 10^{-3}$	$h = 10^{-6}$	pas optimal h
0	1,0	1,0	1,0	1,0	1,0
0,1	0,905	0,900	0,905	0,905	0,905
0,2	0,819	0,809	0,819	0,818	0,819
0,3	0,741	0,726	0,741	0,740	0,741
0,4	0,670	0,649	0,670	0,670	0,670
0,5	0,607	0,578	0,606	0,606	0,607
0,6	0,549	0,511	0,548	0,550	0,548
0,7	0,497	0,447	0,496	0,499	0,496
0,8	0,449	0,384	0,448	0,453	0,447
0,9	0,407	0,320	0,405	0,413	0,404
1,0	0,368	0,250	0,366	0,382	0,366

L'estimation de e_c est obtenue par l'utilisation de la fonction nb.significant.digit(y) qui fournit le nombre de chiffres décimaux significatifs de la fonction y. Ce nombre n_c est un entier obtenu à partir d'une valeur en virgule flottante arrondie par défaut. La valeur de e_c est donc estimée par :

$$e_c = 10^{-n_c+1} \tag{35}$$

L'estimation de e_m est bien connue pour la méthode d'Euler à savoir :

$$e_m = 2|y_1 - y_2| \tag{36}$$

avec y_1 valeur de $y(x_k + h_k)$ intégrée sur l'intervalle $[x_k, x_k + h_k]$ avec le pas h_k,

y_2 valeur de $y(x_k + h_k)$ intégrée sur le même intervalle $[x_k, x_k + h_k]$ deux fois successivement avec le pas $\frac{1}{2}h_k$.

Les programmes utilisés pour résoudre le problème (34) d'une part, en arithmétique en virgule flottante avec des pas d'intégration fixes, d'autre part, avec le pas d'intégration optimal en utilisant le logiciel CADNA, sont présentés au paragraphe 6.

Les résultats obtenus avec ces programmes sont présentés dans le tableau **13**.

Il ressort du tableau **13** que, lorsque le pas h est trop grand ($h = 10^{-1}$) ou trop petit ($h = 10^{-6}$), y n'est pas calculée correctement. Avec ($h = 10^{-3}$) le résultat est correct, mais l'utilisateur ne peut pas connaître *a priori* la valeur de ce pas. Avec CADNA il est possible, grâce à l'utilisation du pas optimal d'obtenir la solution correcte avec sa précision associée et cela quel que soit le pas initial h_0 choisi.

L'exemple présenté est très simple, mais le principe du pas optimal d'intégration a été mis en œuvre dans de nombreuses méthodes, voir [60], [61], [62], [63], [64], [88]. L'apport majeur du logiciel CADNA dans les méthodes numériques approchées est de pouvoir obtenir, en s'affranchissant des paramètres d'intégration (pas d'intégration, degré des polynômes d'approximation dans les méthodes de collocation), la meilleure solution que peut fournir la méthode utilisée ainsi que le nombre de chiffres décimaux significatifs exacts de la solution.

5.4 Fiabilité du logiciel CADNA

L'approche stochastique de l'analyse des erreurs d'arrondi a aussi été mise en œuvre dans deux logiciels, le logiciel PROSOLVER [56] qui implémente d'une manière asynchrone la méthode CESTAC et le logiciel Wongldiff ([65], [66], [67]) qui implémente de manière asynchrone une arithmétique de Monte-Carlo dont l'idée de base est celle de la méthode CESTAC. Dans ce dernier logiciel le nombre N n'est pas fixé et laissé au choix de l'utilisateur. De plus, l'interprétation des résultats est aussi laissée aux bons soins de l'utilisateur.

Ces deux logiciels ne permettent pas de tester la validité de l'hypothèse 2 de la méthode CESTAC en cours d'exécution du programme. Il est donc aisé de les mettre en défaut. Mais est-ce aussi le cas pour CADNA ?

Pour répondre à cette question, imaginons un programme de calcul où l'unique erreur d'arrondi soit la contribution dominante à l'erreur finale. Imaginons également qu'une combinaison particulière d'arrondis aléatoires fasse que la propagation de ces erreurs d'arrondi disparaisse juste sur cet arrondi dominant. Si tel est le cas aucun message n'accompagnera l'affichage du résultat et CADNA sera mis en défaut. Mais, bien sûr, ce cas est exceptionnel et a une probabilité de plus en plus faible de se produire au fur et à mesure que N croît (nombre d'échantillons d'un nombre stochastique discret).

Pour illustrer cela, considérons la résolution du système linéaire donné dans [36] et proposé par J.H. Wilkinson.

Le système est défini par :

$$\mathcal{W}_n \cdot \mathcal{X} = \mathcal{B}$$

avec $\mathcal{W}_n = (w_{i,j})$, $\quad i, j = 1, \ldots, n$ et :

$$\begin{cases} w_{i,i} = 1{,}0 \\ w_{i,j} = -1{,}0 \quad \text{pour} \quad i > j \\ w_{i,j} = 0{,}0 \quad \text{pour} \quad i < j \quad \text{et} \quad j < n \\ w_{i,n} = 1{,}0 \quad \text{pour} \quad i \in [1, n-1] \\ w_{n,n} = \alpha = 0{,}9 \end{cases} \tag{37}$$

Dans ce système, la diagonale et la transposée de la $n^{ième}$ colonne sont (1, 1, ..., 1, α), les éléments de la sous matrice triangulaire supérieure sont nuls exceptés ceux de la $n^{ième}$ colonne et les éléments de la sous matrice triangulaire inférieure sont égaux à – 1. Les n éléments du second membre sont égaux à 1. La solution de ce système qui est bien conditionné est :

$$\begin{cases} x_i^* = -2^{i-1}\dfrac{1-\alpha}{\Delta^*} \quad i = [1, n-1] \\ x_n^* = \dfrac{2^{n-1}}{\Delta^*} \end{cases} \tag{38}$$

avec Δ^* déterminant de la matrice valant $\Delta^* = 2^{n-1} - 1 + \alpha$.

Ce système a été résolu en double précision par la méthode de Gauss avec recherche du pivot maximal par colonne, d'une part en arithmétique en virgule flottante IEEE double précision en arrondi au plus près et d'autre part avec le logiciel CADNA pour $n = 5$, 25, 45. Les résultats sont présentés dans le tableau **14**.

Il ressort du tableau **14** qu'en arithmétique en virgule flottante les solutions fournies pour $n = 25$ et $n = 45$ qui sont affichées avec quinze chiffres décimaux n'en ont en réalité respectivement que 7 et 2 sans que l'utilisateur n'en soit averti. Avec CADNA, seuls les

Tableau 14 – Précision de la solution du système (37) pour différentes dimensions *n*

n	Nombre de chiffres faux sur la solution IEEE 754	Nombre de chiffres exacts estimés par CADNA	Nombre de chiffres exacts par comparaison avec la solution exacte
5	0	15	15
25	8	7	7
45	13	1	2

Tableau 15 – Pourcentage d'échecs en fonction de *n* et *N* dans l'exemple (37)

n	*N* = 3	*N* = 4	*N* = 5	*N* = 6	*N* = 7
5	5	3	2	0	0
25	10	5	3	1	0
45	10	6	4	3	0

chiffres décimaux significatifs exacts sont affichés. Ceux-ci sont en accord avec le nombre de chiffres significatifs décimaux de la solution exacte calculée par les équations (38).

Dans cet exemple, lors de la décomposition *LU* de la méthode de Gauss et du calcul des éléments du second membre $\mathcal{B}$ il n'y a pas de propagation d'erreur puisque les calculs tombent juste. Seul le calcul de la dernière composante $X_n \in \mathbb{F}$ est entaché d'une erreur d'arrondi. Si une combinaison d'arrondis aléatoires fait que les *N* échantillons du nombre stochastique X_n soient égaux, cette erreur d'arrondi disparaît et CADNA est alors mis en défaut. Le tableau **15** présente les pourcentages d'échecs de CADNA en fonction de la dimension *n* et du nombre d'échantillons *N* obtenus sur cet exemple avec une version spéciale du logiciel CADNA qui permet à l'utilisateur de choisir le paramètre *N*.

Il ressort du tableau **15** que *N* doit être égal à 7. Dans ce cas, pourquoi a-t-on choisi *N* = 3 pour l'utilisation générale du logiciel CADNA ? En premier, l'exemple présenté ici n'est absolument pas représentatif des problèmes réels qui sont caractérisés par le fait que le nombre d'arrondis est très grand et participent à l'erreur totale d'arrondi sur les résultats. De plus, il a été montré dans [18], [25], [38] et [45] qu'avec *N* = 3 et une probabilité de 95 % l'équation (16) conduit à une surestimation de un chiffre du nombre de chiffres décimaux significatifs exacts avec une probabilité de 0,054 %. En revanche, la probabilité de sous-estimer la précision de plus de un chiffre est de 29 %. Ainsi, l'estimation de la précision fournie par CADNA peut parfois être pessimiste de un chiffre décimal significatif. En résumé, on peut dire que le logiciel CADNA permet :

– **dans les méthodes finies** :
- de détecter les instabilités numériques,
- de contrôler le bon déroulement du programme,
- d'obtenir tout résultat avec sa précision associée compte tenu, d'une part de la propagation des arrondis de calcul et, d'autre part des incertitudes des données ;

– **dans les méthodes itératives** :
- d'éliminer les ε arbitraires dans les tests d'arrêt,
- d'élaborer des tests d'arrêt optimaux en revenant souvent à la formulation mathématique du problème,
- d'optimiser le nombre d'itérations,
- d'obtenir la précision de la solution ;

– **dans les méthodes approchées** :
- d'estimer le pas optimal dans les méthodes de type différences finies,
- d'optimiser ainsi le nombre de pas de calculs,
- de connaître la précision de la solution calculée ;

– **dans l'étude des systèmes dynamiques chaotiques** :
- de localiser les points où la trajectoire calculée se sépare de la trajectoire exacte,
- de repérer les trajectoires stables.

6. Exemples d'utilisation du logiciel CADNA

Nous présentons ici les programmes correspondants aux exemples donnés dans les chapitres précédents lorsqu'ils n'ont pas été déjà donnés dans ces chapitres. Il s'agit :

– de deux programmes tableaux **16** et **17** relatifs aux méthodes finies ;

– d'un programme relatif aux méthodes itératives (méthode de Jacobi) ;

– d'un programme relatif aux méthodes approchées (méthode d'Euler pour la résolution d'équations différentielles ordinaires).

Tableau 16 – Programmes de calcul de la formule (29)

Sans CADNA	Avec CADNA
program md3	program md3
	use cadna
double precision	type(double_st)t,x,y,z,num,den,m,n,l,a,b,c
t,x,y,z,num,den,m,n,l,a,b,c	
	call cadna_init(–1)
a=3.d+08	a=3.d+08
b=6.d0	b=6.d0
c=5.d0	c=5.d0
n=a*b*c	n=a*b*c
m=a*(b+c)+b*c	m=a*(b+c)+b*c
l=a+b+c	l=a+b+c
x=(b+c)/2.d0	x=(b+c)/2.d0
y=(b*b+c*c)/(b+c)	y=(b*b+c*c)/(b+c)
z=1–(m–n/x)/y	z=1–(m–n/x)/y
num=(m–n/(1–(m–n/z)/(1–(m–n/y)/z)))	num=(m–n/(1–(m–n/z)/(1–(m–n/y)/z)))
den=(1–(m–n/(1–(m–n/y)/z))/ (1–(m–n/z)/& (1–(m–n/y)/z)))	den=(1–(m–n/(1–(m–n/y)/z))/(1–(m–n/z)/& (1–(m–n/y)/z)))
t=1–num/den	t=1–num/den
print*,'t=',t	print*,'t=',str(t)
	call cadna_end()
end program md3	end program md3

Tableau 17 – Programme de calcul du déterminant de la matrice de Hilbert

Programme initial	Programme avec CADNA
program det_hilbert	program det_hilbert
! Calcul du déterminant de la matrice de Hilbert	! Calcul du déterminant de la matrice de Hilbert
! par Gauss sans recherche du pivot max.	! par Gauss sans recherche du pivot max.
! Données précises	! Données imprécises
	use cadna
double precision a(15,15),det	type(double_st) a(15,15),det
	double precision delta
	call cadna_init (–1)
print*,'entrez n'	print*,'entrez n'
read*,n	read*,n
	print*,'entrez eps'
	read*,eps
do i=1,n	do i=1,n
do j=1,n	do j=1,n
a(i,j)=1.d0/(i+j–1)	a(i,j)=1.d0/(i+j–1)
	call data_st(a(i,j),delta)
enddo	enddo
enddo	enddo
call deter(a,15,n,det)	call deter(a,15,n,det)
print*,'determinant=',det	print*,'determinant=',str(det)
	call cadna_end()
end program det_hilbert	end program det_hilbert
subroutine deter (a,namax,n,det)	subroutine deter(a,namax,n,det)
! Calcul d'un déterminant par Gauss	! Calcul d'un déterminant par Gauss
	use cadna
double precision a(15,15), aux, det	type (double_st) a(15,15), aux, det
det=1.d0	det=1.d0
do i=1,n–1	do i=1,n–1
print*,' pivot', i,'=',a(i,i)	print*,' pivot', i,'=',str(a(i,i))
det=det*a(i,i)	det=det*a(i,i)
aux=1.d0/a(i,i)	aux=1.d0/a(i,i)
do j=i+1,n	do j=i+1,n
a(i,j)=a(i,j)*aux	a(i,j)=a(i,j)*aux
enddo	enddo
do j=i+1,n	do j=i+1,n
aux=a(j,i)	aux=a(j,i)
do k=i+1,n	do k=i+1,n
a(j,k)=a(j,k)–aux*a(i,k)	a(j,k)=a(j,k)–aux*a(i,k)
enddo	enddo
enddo	enddo
enddo	enddo
print*,'pivot',i',=',a(i,i)	print*,'pivot',i',=',str(a(i,i))
det=det*a(i,i)	det=det*a(i,i)
return	return
end subroutine deter	end subroutine deter

6.1 Programme de calcul de la formule de Kahan

Selon l'équation (29) dont les résultats sont présentés au paragraphe 5.1. Voir le tableau **16**.

6.2 Programme de calcul du déterminant de la matrice de Hilbert

Voir le tableau **17** au paragraphe 5.1.

6.3 Programme de résolution du système linéaire

Il est présenté au paragraphe 5.2 (équation 33) et résolu par la méthode de Jacobi (voir tableau **18**).

6.4 Programme de résolution du problème de Cauchy

Il est muni de ses conditions initiales et présenté au paragraphe 5.3, équation (34) (voir tableau **19**).

Tableau 18 – Programmes de résolution du système [33] par la méthode de Jacobi

Programme Jacobi initial	Programme Jacobi avec CADNA
program jacobi	program jacobi
	use cadna
	implicit type (single_st) (a–h,o–z)
parameter (eps=3.e–4)	
parameter (ndim=20, niter=1 000)	parameter (ndim=20, niter=1 000)
dimension a(ndim,ndim),b(ndim)	dimension a(ndim,ndim),b(ndim)
dimension x(ndim), y(ndim),xsol(ndim)	dimension x(ndim), y(ndim),xsol(ndim)
	real random
	call cadna_init(–1)
xsol(1)=1.7	xsol(1)=1.7
xsol(2)=–4 746.89	xsol(2)=–4 746.89
xsol(3)=50.23	xsol(3)=50.23
xsol(4)=–245.32	xsol(4)=–245.32
xsol(5)=4 778.29	xsol(5)=4 778.29
xsol(6)=–75.73	xsol(6)=–75.73
xsol(7)=3 495.43	xsol(7)=3 495.43
xsol(8)=4.35	xsol(8)=4.35
xsol(9)=452.98	xsol(9)=452.98
xsol(10)=–2.76	xsol(10)=–2.76
xsol(11)=8 239.24	xsol(11)=8 239.24
xsol(12)=3.46	xsol(12)=3.46
xsol(13)=1 000.0	xsol(13)=1 000.0
xsol(14)=–5.0	xsol(14)=–5.0
xsol(15)=3 642.4	xsol(15)=3 642.4
xsol(16)=735.36	xsol(16)=735.36
xsol(17)=1.7	xsol(17)=1.7
xsol(18)=–2 349.17	xsol(18)=–2 349.17
xsol(19)=–8 247.52	xsol(19)=–8 247.52
xsol(20)=9 843.57	xsol(20)=9 843.57

Tableau 18 – Programmes de résolution du système [33] par la méthode de Jacobi

Programme Jacobi initial	Programme Jacobi avec CADNA
do i=1,ndim	do i=1,ndim
do j=1,ndim	do j=1,ndim
a(i,j)=random()	a(i,j)=random()
enddo	enddo
a(i,i)=a(i,i)+4.9213648	a(i,i)=a(i,i)+4.9213648
enddo	enddo
do i=1,ndim	do i=1,ndim
aux=0.0	aux=0.0
do j=1,ndim	do j=1,ndim
aux=aux+a(i,j)*xsol(j)	aux=aux+a(i,j)*xsol(j)
enddo	enddo
b(i)=aux	b(i)=aux
y(i)=10.0	y(i)=10.0
enddo	enddo
do i=1,niter	do i=1,niter
anorm=0.0	p=.false.
do j=1,ndim	do j=1,ndim
x(j)=y(j)	x(j)=y(j)
enddo	enddo
do j=1,ndim	do j=1,ndim
aux =b(j)	aux =b(j)
do k=1,ndim	do k=1,ndim
if (k.ne.j) then	if (k.ne.j) then
aux=aux–a(j,k)*x(k)	aux=aux–a(j,k)*x(k)
endif	endif
enddo	enddo
y(j)=aux/a(j,j)	y(j)=aux/a(j,j)
if (abs(x(j)–y(j)).gt.anorm) &	if(x(j).ne.y(j)) p=1
p=x(j).ne.y(j)	
anorm=abs(x(j)–y(j))	
enddo	enddo
if (anorm.lt.eps) goto 1	if (p.eq.0) goto 1
enddo	enddo
1 continue	1 continue
write(*,*)'niter=',i–1	write(*,*)'niter=',i–1
do j=1,ndim	do j=1,ndim
aux=–b(i)	aux=–b(i)
do i=1,ndim	do i=1,ndim
aux=aux+a(i,j)*y(j)	aux=aux+a(i,j)*y(j)
enddo	enddo
write(*,*) i,y(i),aux	write(*,*) i,str(y(i)),str(aux)
enddo	enddo
	call cadna_end()
end program jacobi	end program jacobi
function random()	function random()
save nrand	save nrand
data nrand/23/	data nrand/23/
nrand=mod(nrand*5 363+143.1387)	nrand=mod(nrand*5 363+143.1387)
random=2.0*nrand/1387.0–1.0	random=2.0*nrand/1387.0–1.0
return	return
end function random	end function random

Tableau 19 – Programmes d'intégration du système (34) par la méthode d'Euler

Intégration à pas fixe	Intégration avec recherche du pas optimal
program eqdif	program eqdif
	use cadna
implicit none	implicit none
real x0,y0,h0,xlim,f	type(single_st) x0,y0,h0,xlim,f
external f	external f
	call cadna_init(–1)
x0=0	x0=0
y0=1.0	y0=1.0
xlim=1.0	xlim=1.0
print*,'pas d'integration?'	print*,'pas d'integration?'
read*,h0	read*,h0
call euler(f,x0,y0,h0,xlim)	call optstep(f,x0,y0,h0,xlim)
	call cadna_end()
end program eqdif	end program eqdif
subroutine euler(f,x0,y0,h0,xlim)	subroutine optstep(f,x0,y0,h0,xlim)
	use cadna
real x0,y0,h0,xlim,f	type(single_st)x0,y0,h0,h2,xlim,y1,y2,em,ec,f
do while(x0.lt.xlim)	integer icompt
y=y0+h0*f(x0,y0)	2 icompt=0
y0=y	1 icompt=icompt+1
x0=x0+h0	if (icompt.ge.50) then
print*,'x=',x0,'y=',y	print*,'Pas optimal non trouvé'
enddo	stop
end subroutine euler	endif
	! Calcul de y1
	y1=y0+h0*f(x0,y0)
	! Estimation de l'erreur de calcul ec
	ice=nb_significant_digit(y1)
	ec=10**(–ic–1)
	if (ice.gt.7.) ec=10**(–7)
	! Calcul de y2
	h2=h0/2.
	y2=y0+h2*f(x0,y0)
	y2=y2+h2*f(x0+h2,y2)
	! Estimation de l'erreur de méthode em
	em=2.*abs(y1–y2)
	if(em.lt. ec.or. em.eq.0.) em=ec
	3 if(em.eq.ec) then
	x0=x0+h0
	y0=y1
	print*,'x=', str(x0),' y=', str(y1)
	h0=1.5*h0
	if(x0gt.xlim.)stop
	goto 2
	endif
	if(ec.lt.em) then
	h0=h0/3.
	goto 1
	endif
	end subroutine optstep
function f(x,y)	function f(x,y)
	use cadna
real x,y,f	type(single_st) x,y,f
f=exp(x)*y+x*y–(x+1.)*exp(–x)–1.	f=exp(x)*y+x*y–(x+1.)*exp(–x)–1.
end function f	end function f

7. Conclusion

Après avoir défini l'arithmétique approchée (arithmétique en virgule flottante) des ordinateurs et montré les conséquences de cette arithmétique au niveau de chaque opération arithmétique élémentaire, nous avons présenté diverses approches déterministes et probabilistes d'estimation de la propagation des erreurs d'arrondi en calcul scientifique.

Nous avons aussi proposé une méthode d'évaluation de l'influence des incertitudes des données sur les résultats de calcul. Dans le domaine des approches déterministes nous avons d'une part présenté l'analyse régressive des erreurs d'arrondi encore appelée analyse *a posteriori* due à J.H. Wilkinson ainsi qu'un schéma formalisé d'analyse d'erreur du à F.W. Olver, et d'autre part considéré les méthodes basées sur l'arithmétique d'intervalles.

L'essence de l'analyse régressive est de considérer que tout résultat informatique issu d'une suite ordonnée de calculs effectués sur des données, n'est rien d'autre que le résultat exact de cette même suite de calculs effectués sur les mêmes données mais perturbées.

Cette méthode est très intéressante pour la compréhension de la propagation des erreurs d'arrondi lors de l'exécution sur ordinateur d'un algorithme numérique. Elle est par conséquent féconde pour étudier la stabilité numérique des algorithmes. Toutefois, sa mise en œuvre nécessite une étude détaillée souvent longue et délicate de l'algorithme considéré.

Comme, de plus, une majorante de l'erreur d'arrondi est estimée au niveau de chaque opération arithmétique en virgule flottante, la méthode conduit, lors de son utilisation sur un algorithme exécuté sur ordinateur avec des données spécifiques, à une estimation trop pessimiste de l'erreur d'arrondi sur tout résultat fourni par la machine. Comme le formalisme dû à F.W. Olver, l'analyse régressive offre à l'analyste numéricien un cadre d'étude théorique rigoureux. Basé sur cette approche un logiciel nommé PRECISE a été développé dans les années 1990 qui permet de détecter les instabilités dans les algorithmes numériques et ainsi de guider le programmeur dans le choix de la méthode à utiliser.

En ce qui concerne l'arithmétique d'intervalles, son principe consiste à considérer que tout élément de l'ensemble $\mathbb{F}$ des nombres en virgule flottante représentables sur un ordinateur, est l'image d'un intervalle de l'ensemble $\mathbb{R}$ dont tous les éléments après arrondi donnent ce nombre. Ainsi à tout résultat $r \in \mathbb{R}$ d'un algorithme numérique, l'arithmétique d'intervalles associe un intervalle résultat contenant la valeur r.

Toutefois, cette méthode conduit à des intervalles incluant la solution de plus en plus grands au fur et à mesure que le nombre d'opérations nécessaires pour l'obtention du résultat croît. Des études analytiques spécifiques ont été développées pour mettre en œuvre une telle arithmétique car sans précautions de modalité les résultats obtenus par l'arithmétique d'intervalles sont souvent pessimistes.

Dans les années 1970, U. Kulish et les membres de son équipe, après une étude des erreurs d'arrondi dans les ensembles ordonnés ont conclu à la nécessité de faire certains calculs, notamment les produits scalaires, en précision étendue. Ils ont proposé des extensions des langages Fortran, Pascal et C, ainsi qu'une bibliothèque nommée ACRITH permettant de valider les méthodes de calcul scientifique telles que : résolution des systèmes linéaires et celle des équations polynômiales, le calcul des éléments propres des matrices etc. Depuis, des bibliothèques d'arithmétique d'intervalles à précision variable telles MPFI ont été proposées.

Les méthodes déterministes permettent d'étudier la stabilité des algorithmes numériques et de fournir des majorantes de la propagation des erreurs d'arrondi, mais ne répondent généralement pas à la question que se posent les ingénieurs, à savoir : lors de l'exécution d'un programme de calcul scientifique sur ordinateur avec une arithmétique en virgule flottante et des données entachées d'incertitudes, quel est le nombre de chiffres décimaux significatifs

exacts des résultats fournis par la machine ? L'approche stochastique peut répondre à cette question.

Dans cette approche, après avoir formalisé les erreurs absolues d'arrondi dues à chaque opérateur arithmétique en virgule flottante, l'erreur absolue due à la propagation des erreurs d'arrondi sur tout résultat d'un programme scientifique a été établie. De même, l'influence des incertitudes des données sur tout résultat d'un programme de calcul scientifique a été formalisée. À partir de ces résultats, il a été possible de déduire le nombre de chiffres significatifs exacts de tout résultat informatique.

L'idée de base de l'approche stochastique est qu'au cours de l'exécution d'un programme de calcul, certaines erreurs d'arrondi peuvent se compenser. Comme on ne peut pas les contrôler, on les considère comme des variables aléatoires indépendantes équidistribuées. Ainsi tout résultat informatique peut être considéré comme une variable aléatoire dont le nombre de chiffres significatifs exacts dépend des caractéristiques de cette variable (moyenne, écart-type). Pour estimer ces caractéristiques, il est nécessaire d'avoir plusieurs échantillons représentant le résultat informatique. Ces échantillons sont fournis par la méthode CESTAC.

Cette méthode, proposée en 1974 par La Porte et Vignes, a été développée par ce dernier et par les membres de son équipe de recherche. Le principe de la méthode consiste à faire exécuter plusieurs fois en parallèle de manière synchrone le même programme de calcul en propageant différemment les erreurs d'arrondi. Cette propagation différente est obtenue à l'aide d'une arithmétique à arrondi aléatoire qui au niveau de chaque opération arithmétique retient soit le résultat arrondi par défaut ou par excès avec une probabilité $\frac{1}{2}$ pour les nombres qui ne tombent pas juste en machine et bien sûr ne modifie pas ceux qui sont exactement représentés.

Ainsi, pour tout résultat d'un programme de calcul, on obtient plusieurs échantillons représentant le même résultat mais différents en raison des propagations d'erreurs d'arrondi différentes. La moyenne de ces échantillons est considérée comme la meilleure approximation du résultat informatique. L'écart-type étant une approximation de l'erreur, il est alors aisé d'en déduire le nombre de chiffres significatifs exacts du résultat.

L'étude théorique de cette méthode a permis de montrer son efficacité et sa robustesse sous certaines hypothèses qui peuvent être vérifiées en cours de calcul en raison de l'implémentation synchrone de la méthode. Cette implémentation synchrone associée au concept du zéro informatique a permis de développer une nouvelle arithmétique appelée arithmétique stochastique discrète qui possède une grande partie des propriétés de l'arithmétique exacte sur les nombres réels, propriétés qui étaient perdues par l'arithmétique en virgule flottante. L'arithmétique stochastique discrète est mise en œuvre dans le logiciel CADNA.

Le logiciel CADNA permet en cours d'exécution d'un programme d'estimer les effets de la propagation des erreurs d'arrondi et des incertitudes de données sur tout résultat de ce programme. De fait, seuls les chiffres décimaux significatifs exacts sont affichés. Il assure aussi l'auto-validation de la méthode CESTAC et détecte les violations éventuelles des hypothèses qui la sous-tendent. Si tel est le cas l'utilisateur est averti par des messages lui indiquant les types d'anomalies détectées.

L'utilisateur est donc en mesure de faire le débogage numérique du programme.

L'apport de l'utilisation de CADNA en calcul scientifique est important :

– **dans toutes les méthodes finies**, il permet le contrôle des débranchements et fournit la précision des résultats ;

– **dans les méthodes itératives**, il permet d'éliminer les ε arbitraires existant dans les tests d'arrêt et ainsi d'optimiser le nombre d'itérations. Il fournit aussi la précision des résultats ;

– **dans les méthodes approchées**, telles que les méthodes aux différences finies ou les méthodes d'approximation de la solution d'un problème de condition initiale, il permet de déterminer le pas optimal et ainsi d'obtenir la meilleure solution que la méthode choisie et l'ordinateur utilisé peuvent donner. Il fournit la précision de la solution obtenue.

Le logiciel CADNA a été utilisé sur de nombreux algorithmes : algèbre linéaire ([68], [69], [70], [71], [72]), optimisation ([73], [74], [75], [76]), quadratures numériques ([63], [77], [87]), résolution d'équations différentielles, étude de problèmes chaotiques, etc. Il est aussi utilisé par l'industrie dans de nombreux domaines : géologie, géophysique ([78], [79]), acoustique ([80], [81]) , mécanique des fluides, physique, combustion dans les moteurs thermiques [83], [84], [85] et [86].

Des exemples de l'utilisation du logiciel CADNA sont présentés dans le dernier chapitre. En ce qui concerne le temps de calcul, il est clair que la méthode CESTAC et le logiciel CADNA requérant plusieurs fois l'exécution des mêmes opérations avec des arrondis différents ont pour effet d'augmenter ce temps.

Cette augmentation dépend bien évidemment du programme mis en œuvre et de l'ordinateur utilisé (mono ou multi-processeur), de même que de l'implémentation du logiciel CADNA.

Elle peut cependant être estimée à un facteur compris entre 1,5 et 6 fois le temps qui aurait été nécessaire à l'exécution du programme séquentiel d'origine sans validation.

Ce prix à payer pour l'obtention de résultats fiables peut paraître important, mais il faut remarquer ici qu'il s'agit d'une augmentation de temps machine qui se compte en minutes, ou en heures, et non de temps de programmation par des ingénieurs qui se compte en mois ou en années.

Or, la plupart des autres méthodes requièrent des augmentations du temps de calcul encore plus grandes et supposent des modifications importantes du programme initial qui doivent être faites par les programmeurs ce qui explique leur très faible taux d'utilisation, tandis que les modifications minimes du programme nécessitées par l'utilisation de CADNA sont faites très rapidement et peuvent même être faites automatiquement par un pré-processeur.

On peut aussi considérer qu'obtenir des résultats informatiques fiables en particulier dans des applications dites critiques telles que le contrôle du fonctionnement des centrales nucléaires ou le calcul de trajectoires des lanceurs de satellites, n'a pas de prix.

CADNA est mis à disposition des utilisateurs à l'adresse indiquée précisée dans la [Doc. AF 1 470].

En mettant en évidence les conséquences de la propagation des erreurs d'arrondi et l'effet des incertitudes des données sur les résultats numériques fournis par les programmes exécutés sur ordinateur, les auteurs ont voulu insister sur la nécessité absolue de valider ces résultats et ont détaillé une méthode pour le faire.