TP IG3 – Arithmétique flottante

Pour réaliser ce TP vous devrez utiliser un outil appelé le Magnifier qui est disponible grâce à une machine virtuelle présente dans VirtualBox (qui est normalement installé sur votre machine).

Présentation du Magnifier

Le Magnifier est un outil développé par Numalis originellement pour ses besoins internes et qui a été mis à disposition pour l'université de Montpellier. Il permet de tracer le comportement de l'arithmétique flottante à travers un programme écrit dans un langage de script simplifié (le HSL). Ce langage ne supporte que les affectations, les boucles `for`, et les structures conditionnels `if`. Une documentation est disponible en utilisant le menu Help -> User Documentation.

Prise en main du Magnifier

Lancer VirtualBox et démarrer la machine virtuelle : Numalis Magnifier 1.12. Une fois lancée ouvrez le programme Numalis Magnifier sur le bureau. Dans la fenêtre qui s'affiche utiliser File -> Example -> patriot, pour charger l'exemple du missile Patriot HSL. Le premier onglet permet de modifier le code qui est exécuté, après chaque modification cliquer sur Build as new HSL pour obtenir l'analyse de la nouvelle version. Le second onglet permet d'avoir une vue dynamique du programme. Il est possible de cliquer sur chaque point de contrôle du code et voir son comportement en arithmétique flottante. En bas à gauche s'affiche les informations du nœud cliqué.

- P_{value}, correspond à la valeur flottante calculée par la machine
- M_{value}, correspond à la valeur « réelle » calculée avec une librairie de précision arbitraire.
- Rel_{err}, correspond à l'erreur relative, i.e. $\frac{(P_{value}-M_{value})}{M\_value}$
- Raw_{err}, correspond à l'erreur brute, i.e. $P_{value} - M_{value}$

Si vous souhaitez observer le comportement d'un nœud durant l'exécution d'une boucle : sélectionnez le nœud, puis utilisez le menu Graphics -> New from left selection. Dans le nouvel onglet vous pouvez afficher, en simultané, jusqu'à 2 graphiques concernant ce nœud (pensez à activer l'option Create 2nd scale).

Vous pouvez aussi perturber les entrées de votre programme en cliquant sur enable perturbation dans le second onglet. Les perturbations sont soit exprimé en pourcentage, soit en ulp (Unit in the Last Place). Une fois une perturbation effectuée cliquer sur Run Perturbation. Les résultats de la version perturbée apparaissent en bas à droite de l'interface. Dans les onglets contenant les graphiques il est ensuite possible d'observer le comportement entre la version originale et la version perturbée.

Pensez à sauvegarder régulièrement votre travail !

Exercice 1 : Le missile Patriot

1 – Charger l'exemple du missile Patriot et tracer son comportement en erreur brute et en erreur relative.
2 – Trouver une valeur d'incrément qui minimise la dérive numérique du programme
3 – Modifier le programme pour n'utiliser que cette valeur d'incrément mais que celui-ci continue à produire le comportement attendu (i.e. 0.1 puis 0.2, 0.3, etc.). Que constatez-vous en terme d'évolution de l'erreur relative ? Comment expliquez-vous la largeur et la hauteur des zones observées ?
4 – Proposer un programme simulant le comportement du programme initial, mais qui ne contient que des additions et dont l'incrément ne serait pas 0.1. (Indice : penser à utiliser des conditions).
5 – Est-ce que cette dernière version sera plus stable que la version proposée à la question 3 ?

Exercice 2 : Les algorithmes de sommations

Les sommations de nombres peuvent poser des problèmes quand ces nombres sont écrits en virgule flottante. Chaque année de nouveaux algorithmes sont publiés par des scientifiques pour essayer de pallier à ces problèmes.

1 – Rappelez pourquoi les sommations flottantes peuvent poser problème ?
2 – Pour une somme de 3 termes (qu'on nomme a, b, c), de combien de manière différentes peut-on réaliser ce calcul ?
3 – Combien d'entre elles peuvent mener à des résultats différents ?
4 – Et pour 4 termes ? 5 termes ? 6 termes ?

Écrivez un programme qui va réaliser la somme des valeurs suivantes :

```
a = – 513 129 984.0
b = –  32 850 752.0
c =   545 810 688.0
d = –       4 312.8193359375
e = –       3 938.76611328125
f =         2 404.6982421875
```

4 – Essayer d'écrire votre programme pour que la somme de a, b, c, d, e, f soit la plus précise possible ?
5 – Que pouvez-vous dire de la manière optimale de sommer ces 6 termes ?
6 – Et si a, b, d, e étaient tous positifs, quelle serait la forme optimale ? Pourquoi ?

Exercice 3 : Sensibilité d'une fonction

Charger l'exemple concernant la récurrence de Muller. Le programme qui est décrit dans cette exemple est un cas typique de calcul numérique divergent.
1 – Tracer le comportement en terme d'erreur de cette suite

2 – Passer le calcul en double précision (en utilisant le menu haut à gauche : Primitive type). Que constatez-vous ?
3 – Analyser le comportement de la fonction en perturbant ses entrées. Que pouvez-vous dire de sa sensibilité ?

Exercice 4 : Un algorithme de gradient conjugué

L'algorithme de gradient conjugué est une méthode numérique utilisée dans plusieurs industries, dont l'aérospatiale, pour la résolution de problèmes d'optimisation non-contraint.
Le code d'un gradient conjugué est fourni à l'adresse suivante :
http://www.numalis.com/fichiers/temp/conjugue.hsl
À chaque itération le résultat du gradient est stocké dans le tableau `res`. Les premières boucles sont des boucles d'initialisation, la boucle principale est celle définie par l'indice `k`.

1 – Charger le fichier et analyser le comportement en terme d'erreur du tableau res.
2 – Tracer l'origine de l'erreur à partir de ce tableau. Pour cela vous pouvez utiliser les fonctions de gradient dans l'interface (menu Gradient -> Relative error, ou Gradient -> Raw error). Un gradient permet de visualiser l'intensité d'une donnée sur l'ensemble des nœuds. Plus la couleur est rouge foncé ou bleu foncé, plus la valeur est intense.
3 – Que pourriez-vous proposer pour améliorer ce programme ?

Exercice 5 : Les polynômes

Les polynômes sont des fonctions mathématiques qui sont composés d'additions de multiplications (et parfois de divisions).

1 – Pourquoi est-ce une bonne chose en termes de calcul avec des nombres flottants ?

On s'intéresse d'abord à un polynôme P (très simple) décrit ci-dessous.

$$P(x) = a + b * x + c * x^2 + d * x^3 + e * x^4 + f * x^5$$

Ecrivez ce polynôme en HSL avec les valeurs suivantes :
a = 1.0
b = 2.0
c = 3.0
d = 4.0
e = 5.0
f = 6.0
et x = 0.1

2 – Essayer d'expliquer comment la manière dont vous l'avez écrit sera réalisée par la machine ? (N'utilisez pas la fonction `pow`)

3 – Comment pourriez-vous changer la manière d'écrire ce polynôme ?
4 – En utilisant les réponses aux exercices précédent essayer d'améliorer la précision de ce polynôme.

Parmi l'ensemble de tous les polynômes possibles certains polynômes sont utilisés pour calculer les valeurs de fonctions plus complexes. C'est le cas par exemples des polynômes d'interpolation dit de « développements limités » qui permettent d'approximer, jusqu'à un certain point, des fonctions complexes comme : log, exp, cos, sin, tan, etc. Nous allons nous intéresser à ce type de polynômes sur 2 fonctions très largement utilisées : logarithme et exponentielle.

5 – Rappelez pourquoi ces deux fonctions peuvent poser problèmes quand elles sont calculées sur ordinateur ?

a) Développement limité de logarithme

Le développement limité de logarithme proche de zéro s'écrit de la manière suivante :

$$\ln(1+x) \approx \sum_{i=1}^{n} (-1)^n \times \frac{x^n}{n}$$

Avec cette méthode plus la valeur n augmente plus le polynôme calcule une valeur proche de ln(1+x).

Par exemple si n=4 le développement limité donne l'expression suivante :

$$-x + \frac{x^2}{2} - \frac{x^3}{3} + \frac{x^4}{4}$$

6 – Ecrivez en HSL (sans la fonction `pow`) le développement limité de logarithme quand n=4 et essayer de l'évaluer plusieurs valeurs à virgule entre 0 et 4. Que constatez-vous ?
7 – Essayer de l'évaluer sur des valeurs exactes et des valeurs dont vous savez qu'elles ne sont pas représentables en flottant. Que constatez-vous ?

b) Développement de l'exponentielle

Le développement limité de la fonction exponentielle proche de zéro s'écrit de la manière suivante :

$$e^x \approx \sum_{i=0}^{n} \frac{x^n}{n!}$$

Par exemple si n=4 le développement limité est :

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}$$

8 – Comme précédemment écrivez en HSL (sans la fonction `pow`) le développement limité de l'exponentielle quand n=4 et essayer de l'évaluer plusieurs valeurs à virgule entre 0 et 4.

9 – Essayer de l'évaluer sur des valeurs exactes et sur des valeurs dont vous savez qu'elles ne sont pas représentables en flottant ?

10 – (Optionnel) La fonction logarithme est l'inverse de la fonction exponentielle. Essayer de les composer pour vérifier si leurs développements limités le sont bien aussi ? Quel impact la précision représente ?