Implantation de fonctions mathématiques avec Sollya & Gappa

Christoph Lauter Guillaume Melquiond

26 mars 2013

1 Prise en main des outils

Pour commencer, nous allons nous faire la main avec les deux outils que nous utiliserons, Sollya et Gappa.

1.1 Sollya

Sollya peut être utilisé en deux modes différents :

- En mode interactif où on donne des commandes au prompt Sollya et où on regarde les affichages faits par l'outil. Ce mode est approprié pour de courtes expérimentations avec une fonction. En revanche, l'utilisation des structures de contrôle est moins aisée en interactif.
- En mode *exécution de scripts* qu'on rédige dans un éditeur de texte et qu'on fait exécuter par l'outil.

Pour le travail en TP, nous vous recommandons de lancer Sollya d'abord en mode interactif mais de passer assez rapidement en mode script.

Pour lancer Sollya en mode interactif, il faut l'exécuter au prompt shell à travers l'outil `rlwrap` :

```
$ rlwrap -A sollya
>
```

Pour exécuter un script écrit dans un éditeur de texte, on lance Sollya sur le script [1] ; l'utilisation de `rlwrap` n'est pas nécessaire dans ce cas :

```
$ emacs monscript.sollya &
$ sollya monscript.sollya
2
Coucou
```

1. Lancer Sollya en mode interactif et jouer un peu avec ce mode *calculette* :

1. L'utilisation de scripts exécutables avec un shebang au début est également possible avec Sollya.

- Calculer $1+1$, $1+2^{63}$, $\exp(5)$, $\sin(\exp(0)-1)$, $\sin(1/13)$. Qu'observe-t-on ?
- Définir une fonction $f(x) = \sin(x)$ et l'évaluer en 0, π, 17. Essayer également l'évaluation sur un intervalle comme $[-1;1]$, $[-2^{-5};2^{-5}]$.
- Poser des questions booléennes à Sollya et comprendre ce que répond l'outil :

$$\begin{aligned} f(5) &\overset{?}{<} f(17) \\ f(6\,\pi) &\overset{?}{<} 1/7 \\ f(6\,\pi) &\overset{?}{\leq} 0 \\ f(6\,\pi) &\overset{?}{\leq} \frac{\log_2(13)}{\log_2(17)} - \frac{\log(13)}{\log(17)}. \end{aligned}$$

- Afficher la fonction f sur les domaines $[-5;5]$ et $[-1/16;1/16]$ en se servant de la commande Sollya `plot`. Approcher f par $p(x) = x - 1/6\,x^3 + 1/120\,x^5 - 1/5040\,x^7$ et afficher les erreurs absolue $\delta = p - f$ et relative $\varepsilon = p/f - 1$ sur $[-1/16;1/16]$.

2. Se faire la main avec le système d'aide de Sollya intégré : faire `help help;` au prompt Sollya, ensuite essayer `help` *`macommande`*`;`
3. Passer en mode *exécution de script*. Rédiger un court script Sollya dans un éditeur de texte qui affiche `Hello world!`. Exécuter le script avec Sollya.
4. Changer le script pour y définir la fonction $f(x) = \cos(x)$ et le domaine $I = [-2^{-5};2^{-5}]$. Ensuite, utiliser une boucle `for` pour calculer tous les polynômes de Remez approchant f en erreur absolue sur le domaine I de degré $n = 2$ à $n = 10$. Calculer également la norme sup de l'erreur $\|p - f\|_\infty^I$.
5. Modifier le script pour calculer, pour la fonction et le domaine donnés, le polynôme de Remez de degré minimal pour lequel l'erreur absolue est bornée par 2^{-24}. On a besoin des commandes `if ... then` et `while` pour cela.
6. Écrire une fonction en C qui implante $\log_2(x)$ de façon très naïve :

```
double mylog2(double x) {
  return 1.4426  * (x - 1.0) -
         0.72135 * (x - 1.0) * (x - 1.0);
}
```

Rajouter l'inclusion du fichier d'en-tête `tp.h` et compiler avec

```
$ gcc -O3 -std=c99 -fPIC -c log2-naif.c
$ gcc -shared -o log2-naif.so log2-naif.o
```

Lancer une session Sollya interactive et exécuter la commande :

```
> externalplot("./log2-naif.so", relative, log2(x),
                [1 - 1/16, 1 + 1/16], 15, perturb);
```

Se servir de la commande `help externalplot;` pour comprendre le graphe affiché. Comparer le graphe avec le result d'un calcul de

$$\left\| \frac{1.4426\,(x-1) - 0.72135\,(x-1)^2}{\log_2(x)} - 1 \right\|_\infty^{[1-1/16;1+1/16]}$$

fait en Sollya avec la commande `supnorm`.

1.2 Gappa

Gappa n'est pas interactif et ne peut vérifier qu'une seule formule logique à la fois. Donc autant l'utiliser associé à un éditeur de texte :

```
$ emacs monscript.g &
$ gappa monscript.g
Results:
1 + 1 = 2
```

1. Demander à Gappa quel est l'intervalle de l'expression $x - 1$ pour $x \in [1 - 1/16, 1 + 1/16]$.
2. Demander quel est le flottant `binary64` le plus proche de 1.4426. Vérifier qu'il est différent de celui `binary32`.
3. Vérifier que le calcul $\circ(x - 1)$ est en fait exact pour $x \in [1 - 1/16, 1 + 1/16]$.
4. Calculer l'erreur absolue (quantification et arrondi) de l'évaluation polynomiale de la fonction naïve `mylog2` ci-dessus.
5. Vérifier à l'aide d'un découpage en sous-intervalles que la borne obtenue est (vraisemblablement) optimale.
6. Essayer de calculer l'erreur relative. Trouver pour quelle raison Gappa échoue. Modifier le code C pour corriger ce défaut. Calculer l'erreur relative.
7. Introduire dans le script Gappa l'erreur de troncation calculée par Sollya entre le polynôme et $\log_2(x)$. En déduire l'erreur globale de cette implantation naïve.

2 Implantation du logarithme en base 2

Le but de cet exercice est d'implanter le logarithme en base 2. On ciblera une précision d'au moins 45 bits : le résultat y de la fonction pour une entrée x `binary64` doit vérifier

$$\exists \varepsilon,\ |\varepsilon| \leq 2^{-45} \wedge y = (\log_2 x)(1 + \varepsilon).$$

Le fichier `tp.h` contient quelques fonctions C utiles pour cette implantation. En particulier, `decomposeDouble` prend un flottant x et renvoie un exposant e et une mantisse m flottante entre 1 et 2 tels que $x = m \cdot 2^e$.

2.1 Logarithme sans table

Une réduction d'argument triviale serait $\log_2 x = e + \log_2 m$. Malheureusement, pour m proche de 2 et e valant -1, il y aurait une annulation catastrophique lors de la reconstruction entre la valeur approchée de $\log_2 m$ et e. La solution pour contourner ce problème est de ne pas choisir m entre 1 et 2, mais entre $1/\sqrt{2}$ et $\sqrt{2}$.

2.1.1 Approximation polynomiale en Sollya

1. Calculer par la méthode de Remez une approximation polynomiale de $\log_2(1+r)$ pour r entre $1/\sqrt{2}-1$ et $\sqrt{2}-1$ qui cible l'erreur relative. Donner une norme supérieure sur cette erreur. Trouver le degré minimal du polynôme d'approximation tel que cette erreur soit inférieure à 2^{-45}.
2. Calculer le polynôme d'approximation avec la commande `fpminimax` et donner la norme supérieure sur l'erreur relative.

Questions annexes :
- Pourquoi faire une approximation polynomiale de $\log_2(1+r)$ et non pas de $\log_2(m)$?
- Pourquoi s'intéresser à l'erreur relative et non pas à l'erreur absolue de l'approximation polynomiale ?
- Comment interpréter la borne inférieure de la norme supérieure ?

2.1.2 Code C et Gappa

1. Écrire le code C de la fonction, compiler en `.so` et tester à l'aide de la commande `externalplot` de Sollya.
2. Calculer l'erreur d'arrondi de l'évaluation polynomiale.
3. Calculer l'erreur relative globale prenant en compte l'approximation polynomiale et la reconstruction. Il pourra être nécessaire d'étudier à part le cas $e = 0$.

2.2 Logarithme avec table

Cette fois, le polynôme d'approximation de $\log_2(1+r)$ ne sera plus choisi entre $1/\sqrt{2}-1$ et $\sqrt{2}-1$ mais entre -2^{-8} et 2^{-8} afin de faire chuter son degré à précision égale. La nouvelle réduction d'argument pour $1 \leq m < 2$ est

$$\log_2(x) = (e + \alpha_i) - (\log_2(c_i) + \alpha_i) + \log_2(1 + (c_i m - 1))$$

avec $i = \lfloor 2^7(m-1) + 0.5 \rfloor$ et $c_i m \simeq 1$.

2.2.1 Réduction d'argument

1. Supposons tout d'abord que tous les α_i valent zéro. Quelles valeurs de e et $\log_2(c_i)$ vont provoquer une annulation catastrophique ? Comment choisir les valeurs de α_i pour éviter ça ?
2. Calculer l'ensemble des valeurs prises par l'indice i pour m parcourant l'intervalle $1 \leq m < 2$. Calculer l'intervalle M_i des valeurs de m se réduisant en une valeur de i donnée.
3. Proposer une constante c_i telle que $c_i m \simeq 1$ pour toutes les valeurs $m \in M_i$.
4. Calculer l'intervalle R_i des valeurs prises par $c_i m - 1$ pour $m \in M_i$. Calculer l'intervalle $R = \bigcup_{0 \leq i \leq 128} R_i$.

Question annexe :
- Pourquoi est-il souhaitable d'avoir $c_0 = 1$ et $c_{128} = 1/2$?

2.2.2 Code C

1. Calculer un polynôme d'approximation de $\log_2(1+r)$ sur l'intervalle R à l'aide de Sollya.
2. Écrire le code C de la fonction, compiler et tester. Cette fonction s'appuiera sur deux tables : l'une contenant les valeurs de c_i, l'autre les valeurs de $\alpha_i + \log_2(c_i)$.
3. Calculer l'erreur relative globale prenant en compte la réduction d'argument, l'approximation polynomiale et la reconstruction. Il pourra être nécessaire d'étudier à part les cas $e + \alpha_i = 0$, $i = 0$ et $i = 128$ en Gappa.

2.2.3 Réduction d'argument améliorée

L'approche précédente ne permet pas d'atteindre une précision finale suffisante sur l'erreur globale à cause de l'imprécision qui apparaît lors du produit $c_i m$. Le fichier `tp.h` fournit une fonction `cutDouble` qui permet de transformer un flottant x en deux flottants x_h et x_l tels que $x = x_h + x_l$, x_l est négligeable devant x_h, x_h s'écrit avec 21 bits significatifs et x_l s'écrit avec 32 bits significatifs.

Cette fois, le polynôme ne sera donc pas évalué en $c_i m - 1$ mais en $(c_i m_h - 1) + c_i m_l$.

1. Combien de bits significatifs peuvent avoir les constantes c_i au plus pour que les opérations $c_i \times m_h - 1$ et $c_i \times m_l$ figurant dans la réduction d'argument soient exactes ?

2. Recalculer les intervalles R_i, le polynôme d'approximation, et les valeurs de la table. Écrire le code C de la fonction et la tester.
3. Recalculer l'erreur globale.