Synthèse de code pour l'évaluation de fonctions spéciales

Thématique :	arithmétique en virgule flottante, calcul formel, calcul numérique rigoureux
Cadre :	LIX, École polytechnique
Encadrant :	Marc Mezzarobba <marc@mezzarobba.net>
Prérequis :	analyse réelle, programmation en C et en Python

English Summary

This project is part of a long-term effort to automate the machine-precision floating-point implementation of special functions (mathematical functions slightly less ubiquitous than elementary functions like sin, cos, log, yet "common enough to deserve a name") and other mathematical functions appearing in specialized applications. Code generation has the potential to allow techniques developed for elementary functions to be applied at low cost to a much wider class of functions. This should provide faster and more accurate implementations that can also be better tailored to each application.

The intern will focus on Bessel functions, a family of special functions of central importance in mathematical physics. The aim of the project will be to automate the techniques used in a state-of-the-art human-written implementation of Bessel functions and integrate the result in a code generation framework under development.

Présentation générale du domaine

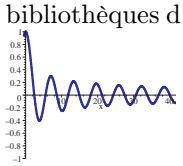
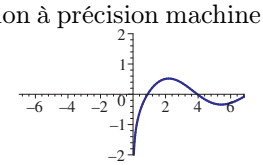
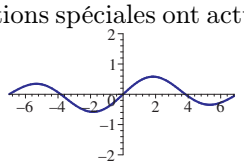
Ce stage se situe dans le domaine de l'arithmétique des ordinateurs. Il s'agit de la branche de l'informatique qui s'intéresse à la représentation des nombres (entiers, réels...) sur ordinateur et aux opérations de base dessus, comme les opérations arithmétiques ($+$, $\times$...) et l'évaluation de fonctions élémentaires (exp, ln, sin...). Le développement d'implémentations à la fois efficaces et précises de fonctions mathématiques, comme celles disponibles dans les bibliothèques standard des langages de programmation, demande une certaine expertise et un temps important. C'est pourquoi, depuis quelques années, des chercheurs se sont attachés à développer des outils pour faciliter ce travail. Il devient aujourd'hui réaliste de développer des *générateurs de code* capables de produire automatiquement des programmes d'évaluation numérique en fonction de divers paramètres au lieu d'écrire ces programmes « à la main ». Outre l'arithmétique des ordinateurs, les techniques mises en œuvre relèvent du calcul formel, du calcul numérique rigoureux et de la compilation.

Objectifs du stage

Les fonctions de Bessel (comme J_0, Y_0 et J_1 ci-dessous), solutions de l'équation différentielle

$$x^2\,y''(x) + x\,y'(x) + (x^2 - \nu^2)\,y(x) = 0,$$

constituent une famille classique de fonctions spéciales (c'est-à-dire de fonctions classiques mais moins fréquentes que les fonctions élémentaires), largement utilisée notamment en physique mathématique. Les bibliothèques d'évaluation à précision machine de fonctions spéciales ont actuellement

des années de retard sur les meilleures bibliothèques de fonctions élémentaires en termes de précision, d'efficacité et de rigueur. Cependant, les progrès de la synthèse de code spécialisée permettent

d'espérer réduire cet écart. L'objet de ce stage est de comprendre comment faire cela dans le cas des fonctions de Bessel.

Le but principal sera de concevoir et implémenter un générateur automatique d'implémentations à précision machine de fonctions de Bessel, en s'appuyant sur des outils tels que Metalibm[1], Sollya[2], Arb[3] et SageMath[4]. Ce générateur prendra en entrée des informations telles que le paramètre ν de la fonction à produire, la précision requise, ou encore le domaine à couvrir, et devra produire un programme d'évaluation en langage C de qualité aussi proche que possible de celui qu'écrirait un expert.

La piste la plus prometteuse pour ce faire consiste à automatiser l'approche mise en œuvre par J. Harrison (Intel) dans une des meilleures implémentations existantes des fonctions de Bessel. Sommairement, il s'agit de combiner des approximations polynomiales précalculées, des développements locaux de la fonction au voisinage de ses zéros, et une utilisation astucieuse de développements asymptotiques de la fonction elle-même ainsi que de ses zéros.

De nombreuses extensions pourront être envisagées suivant les centres d'intérêt du ou de la stagiaire, par exemple en vue de prouver formellement la correction du code produit, ou encore de généraliser les techniques à d'autres familles de fonctions spéciales. À plus long terme, le but est de développer les techniques algorithmiques et les outils logiciels nécessaires pour produire automatiquement des bibliothèques complètes de fonctions spéciales adaptées aux besoins de chaque application.

Bibliographie

[1] Nicolas Brunie, Florent de Dinechin, Olga Kupriianova, and Christoph Lauter. Code generators for mathematical functions, 2015. URL: http://hal.upmc.fr/hal-01084726.

[2] Digital library of mathematical functions, 2010. URL: http://dlmf.nist.gov/.

[3] A. Gil, J. Segura, and N. M. Temme. *Numerical methods for special functions*. SIAM, 2007.

[4] John Harrison. Fast and accurate Bessel function computation. In Javier D. Bruguera, Marius Cornea, Debjit DasSarma, and John Harrison, editors, *ARITH 19*, pages 104–113, Portland OR, 2009. IEEE. URL: http://www.cl.cam.ac.uk/~jrh13/papers/bessel.html.

[5] Christoph Lauter and Marc Mezzarobba. Semi-automatic floating-point implementation of special functions. In Jean-Michel Muller, Arnaud Tisserand, and Julio Villalba, editors, *ARITH 22*, Lyon, France, 2015. IEEE. URL: https://hal.archives-ouvertes.fr/hal-01137953.

1. https://github.com/metalibm/metalibm
2. https://www.sollya.org/
3. http://arblib.org/
4. http://arblib.org/