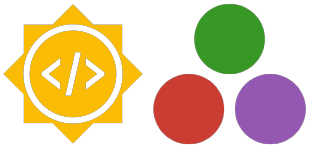


GSoC 2025: Special Functions in The Julia Programming Language

Ahmed Y. Kadah[1]
Oscar D. Smith[2]
[1] Student, University of Science and Technology in Zewail City
2 Mentor, JuliaHub

Abstract

In this project, we will work on implementing high speed and high accuracy implementations of various special functions, as well as optimizing and improving existing implementations in Julia. Our work will mostly be on the SpecialFunctions.jl Julia library but will touch on some of the issues in the HyperGeometricFunctions.jl library as well.

Improve the abstract

Introduction

About me

Hi there! I am Ahmed Yasser Kadah, a third year Communications and Information Engineering student at the University of Science and Technology in Zewail City, Giza, Egypt. I recently took an advanced course in partial differential equations and special functions, so when I saw this project I thought it would be interesting to follow along this path and study the topics further when it comes to real world applications and make something that would genuinely benefit others. This is my first time contributing to an open source project and just recently made my first contribution with a pull request for a Julia implementation of the Float64 erf(x) (will be explained later). [1]

Background on Special Functions

Special functions are a class of mathematical functions that arise in the solutions of complex physical, engineering, and mathematical problems.[2] These functions are typically defined by specific properties or satisfy particular differential equations, and they include some of the most well-known functions in mathematics, such as the error function, Gamma function, and Bessel functions.

This class of functions serve a unique purpose in many different applications, as they have properties which can be utilized to solve highly complex problems which otherwise have no analytic solutions, or even assist in numerical computation.

Due to the nature and importance of this topic, there already exists a large number of implementations in other languages such as Wolfram Mathematica, MATLAB, and many more. However, the number of memory efficient and open source implementations are somewhat limited to languages like C. Hence, our aim here is not to reinvent the wheel, but refine it.

SpecialFunctions.jl and HyperGeometricFunctions.jl are Julia libraries which aim to provide Julia implementations/interfaces for special functions. While HyperGeometricFunctions.jl is purely implemented in julia, SpecialFunctions.jl makes use of external libraries like openlibm and openspecfun which adds a layer of overhead and reduces the efficiency.

Goals

The goals of the project are to (1) implement some of the functions which currently use external library calls in Julia, (2) work on fixing some of the existing issues in the github repositories, (3) optimizing poorly optimized implementations, (4) explore and add missing functions.

The implementations of functions will make use of different methods of numerical approximation methods like polynomial/rational approximation. Some of the important tools include:

1. Remez.jl, a library for constructing minimax polynomials and rational functions
2. Sollya, another powerful tool for constructing polynomials but with a less convenient interface. Will be used as the reference for the performance of Remez.jl and possibly some contributions.

Deliverables and Timeline

Deliverables

- Fully implemented **error function** and it's varieties [4]
  - Error Function: $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t$ for Float32, and BigFloat (Float64 has been implemented as said previously), for real as well as complex inputs.
  - Complementary Error Function: $\operatorname{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t$ for Float32, Float64, and BigFloat, for real as well as complex inputs.
  - Scaled Complementary Error Function: $\operatorname{erfcx}(x) = e^{x^2} \operatorname{erfc}(x)$ for Float32, Float64, and BigFloat, for real as well as complex inputs.
  - Imaginary Error Function: $\operatorname{erfi}(x) = -i \operatorname{erf}(ix)$ for Float32, Float64, and BigFloat, for real as well as complex inputs.
  - Dawson Function: $D(x) = \frac{\sqrt{\pi}}{2} e^{-x^2} \operatorname{erfi}(x)$ for Float32, Float64, and BigFloat, for real as well as complex inputs.
  - Faddeeva Function: $w(z) = e^{-z^2} \operatorname{erfi}(z) = \operatorname{erfcx}(-iz)$ for Float32, Float64, and BigFloat, for complex inputs.
- At least 15 issues resolved on **SpecialFunctions.jl**
- At least 5 issues resolved on **HyperGeometricFunctions.jl**
- Implementation of Appel Hypergeometric Function [17]

Reword this

Add Part about compatibility of julia implementations with auto diff, and explain what autodiff is

Cover more ideas and issues

Go into more detail about SpecialFunctions.jl and HyperGeometricFunctions.jl and what exactly is going to be done, then add the specifics in the deliverables

give proper description of each

Break down each deliverable into what exactly is being delivered, ex. the definition of the error function, what are the targeted data types, etc.

– F_1 is defined for $|x| < 1$, $|y| < 1$ by the series

$$F_1(a, b_1, b_2; c; x, y) = \sum_{m,n=0}^{\infty} \frac{(a)_{m+n}(b_1)_m(b_2)_n}{(c)_{m+n}m!n!}x^m y^n$$

where $(q)_n$ is the Pochhammer symbol.

– F_2 is defined for $|x| + |y| < 1$ by the series

$$F_2(a, b_1, b_2; c_1, c_2; x, y) = \sum_{m,n=0}^{\infty} \frac{(a)_{m+n}(b_1)_m(b_2)_n}{(c_1)_m(c_2)_n m!n!}x^m y^n$$

– F_3 is defined for $|x| < 1$, $|y| < 1$ by the series

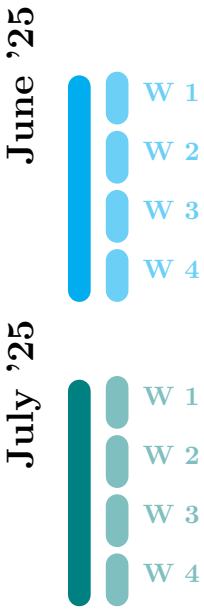
$$F_3(a_1, a_2, b_1, b_2; c; x, y) = \sum_{m,n=0}^{\infty} \frac{(a_1)_m(a_2)_n(b_1)_m(b_2)_n}{(c)_{m+n}m!n!}x^m y^n$$

– F_4 is defined for $|x|^{1/2} + |y|^{1/2} < 1$ by the series

$$F_4(a, b; c_1, c_2; x, y) = \sum_{m,n=0}^{\infty} \frac{(a)_{m+n}(b)_{m+n}}{(c_1)_m(c_2)_n m!n!}x^m y^n$$

Timeline

Finish Timeline



Expected Results

Complete, functional, high efficiency and accuracy, julia implementations of the following:

Finish this or make it into expected difficulties or things along that line

References

[1] https://github.com/JuliaMath/SpecialFunctions.jl/pull/491

[2] https://en.wikipedia.org/wiki/Special_functions

[3] https://en.wikipedia.org/wiki/Hypergeometric_function

[4] https://en.wikipedia.org/wiki/Error_function

[5] https://github.com/JuliaMath/SpecialFunctions.jl

[6] https://github.com/JuliaMath/HypergeometricFunctions.jl

[7] https://reference.wolfram.com/language/guide/SpecialFunctions.html.en

[8] https://reference.wolfram.com/language/guide/HypergeometricFunctions.html

[9] https://www.mathworks.com/help/matlab/special-functions.html

[10] https://www.mathworks.com/help/symbolic/sym.hypergeom.html

[11] https://docs.scipy.org/doc/scipy/reference/special.html

[12] https://www.gnu.org/software/gsl/

[13] https://github.com/JuliaDiff/ForwardDiff.jl

[14] https://github.com/simonbyrne/Remez.jl

[15] https://www.sollya.org/

[16] https://en.wikipedia.org/wiki/Remez_algorithm

[17] https://en.wikipedia.org/wiki/Appell_series

Finish all citations