SOEN 6011: SOFTWARE ENGINEERING PROCESSES
Gamma Function Scientific Calculator

Github: https://github.com/aishangchixiang/SOEN6011
Wenshu Li 40203982

August 5, 2022

Contents

0.1 Introduction

0.1.1 Discription

In mathematics, the gamma function is one commonly used extension of the factorial function to complex numbers[1], is a meromorphic function defined in the complex range, usually written so that negative integers and 0 are its first order poles.There are various definitions of the gamma function, we selected three of them here:

- For any positive integer n,

$$\Gamma(n) = (n-1)!$$

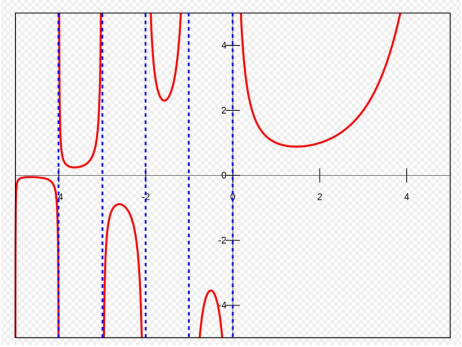
- Derived by Daniel Bernoulli, for complex numbers with a positive real part, the gamma function is defined via a convergent improper integral:

$$\Gamma(z) = \int_0^{+\infty} x^{z-1} e^{-x} \mathrm{d}x, \Re(z) > 0$$

- The gamma function on real number field is defined as:

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} \mathrm{d}t \, (x > 0)$$

Figure 1: Gamma function



0.1.2 Domain

All complex numbers except those whose real part are non-positive integers.

$$C/\{n \in Z, n \leq 0\}$$

0.1.3 Co-Domain

All real numbers excluding zero.

$$(-\infty, 0) \cup (0, +\infty)$$

0.1.4 Properties

One of the important functional equatuations for the gamma fucntion is Euler's reflection formula:

$$\Gamma(1-z)\,\Gamma(z) = \frac{\pi}{\sin \pi z}, z \notin Z$$

0.1.5 Perticular Values

It is easy to find some special values for the gamma function on the web, which can be used as subsequent test cases. Including up to the first 20 digits after the decimal point, some of particular values of the gamma function are[2]:

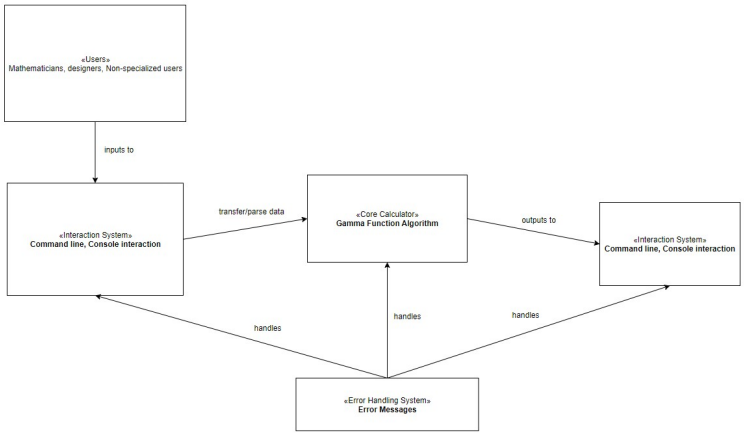
- $\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi} \approx +1.77245385090551602729$
- $\Gamma(1) = 0! = +1$
- $\Gamma\left(\frac{3}{2}\right) = \frac{\pi}{2} \approx +0.88622692545275801364$
- $\Gamma(2) = 1! = +1$
- $\Gamma\left(\frac{5}{2}\right) = \frac{3\sqrt{\pi}}{4} \approx +1.32934038817913702047$
- $\Gamma(3) = 2! \approx +2$
- $\Gamma\left(\frac{7}{2}\right) = \frac{15\sqrt{\pi}}{8} \approx +3.32335097044784255118$
- $\Gamma(4) = 3! = +6$

0.1.6 Stakeholders and Context of Use Model

- User1: Mathematicians and researchers in the fields of calculus, mathematical analysis, statistics.
- User2: Algorithm designers, researchers,engineers in the field of scientific computing in computer and software engineering.
- User3: People with low maths and programming background who need to obtain the values of gamma function.

As can be seen from Figure 2, the user can input the data he wants to input through the interactive system, such as the value of some independent variables, and then the data is converted into data that can be used by the core algorithm part, and the result is calculated and then presented to the user through the interactive system, which includes error handling, such as the user inputting illegal characters that can cause the calculator not to recognize, etc.

Figure 2: Context of use model



0.2 Requirements

0.2.1 Assumptions

- A1: We shall only consider the positive value as input value,
- A2: We shall only consider the real number as input value, ignore the imaginary part.
- A3: We can accept a certain error in the calculation results, but it needs to be within a certain range.
- A4: The error of the calculation result should be within a certain range.

0.2.2 Functional Requirements

- FR1-A3: When the user enters a value that is a non-numeric character, we will give the prompt "Error!".
- FR2-A1: When the user enters a negative value, the function will return "Negative input is not allowed.".
- FR3-A1: When the function value exceeds the maximum the value of a double type variable or equals to zero, the function will return "Infinity".
- FR4-A4: When the input value is a positive integer, the error between the output value and the real value shall be less than 0.001, individual errors greater than 0.001 are allowed, but must be less than 0.1.

0.2.3 Non-functional Requirements

- NFR1: The calculator project shall be maintainable, the code is saved by the github repository, and all functions in the program can be modified and maintained.
- NFR2: The calculator project shall be portable, the project can be run on a computer with a java environment and does not require high computer hardware and computing power.
- NFR3: The calculator project shall be usable, it can be used by different users to calculate the value of the gamma function.

0.3 Algorithm

0.3.1 Integrals Method

Description

The first algorithm is based on the Gamma function formula derived by Daniel Bernoulli, using an approximating integrals method called trapezoidal rule[3]. It is used for initial value problems. In calculus, the trapezoidal rule is a technique for approximation the definite integral. This algorithm involves two sub-functions. One is the power function, which given base and exponent as double type arguments, returns $base^{exponent}$. And the exp function, which given exponent as double type argument, returns $e^{exponent}$.

0.3.2 Lanczos Approximation

Description

The second algorithm is based on Lanczos approximation[4]. In mathematics, the Lanczos approximation is a method for computing the gamma function numerically, published by Cornelius Lanczos in 1964. It is a practical alternative to the more popular Stirling's approximation[5] for calculating the gamma function with fixed precision. This algorithm also involves power function, exp function, and sqrt function which given one double type argument, returns square root.

0.3.3 Pseudocode

Pseudocode generally does not actually obey the syntax rules of any particular language; there is no systematic standard form[6]. So, according to the rules of java programming language and the basic specification of pseudo-code, the following pseudo-code is written, which contains the code of Algorithm 1 and Algorithm 2, which involves some sub-functions such as pow(), sin(), ln(), etc. The principle of their implementation is to replace the original expression with a level form represented only by addition, subtraction, multiplication and division,

according to the Taylor series[7], which is implemented by a loop or recursive method. The formula is as follows:

- The power function:

$$pow(a,x) = a^x = e^{x \ln a} = 1 + \frac{x \ln a}{1!} + \frac{(x \ln a)^2}{2!} + \frac{(x \ln a)^3}{3!} + \cdots \ (-\infty < x < \infty)$$

- The logarithm function:

$$\ln(x) = \ln(\frac{1+y}{1-y}) = 2(\frac{1}{1} + \frac{1}{3}y^2 + \frac{1}{5}y^4 + \frac{1}{7}y^6 + \frac{1}{9}y^8 + \cdots) \ (-\infty < x < \infty)$$

- The sin function:

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \ (-\infty < x < \infty)$$

Algorithm 1 Calculate Gamma function using integrals method

Input: Input value of x
Output: Output value of Gamma function

function $f_y(x)$
    return $s^{x-1} * e^{-s}$
end function

function $gamma(x)$
    $output \leftarrow Error$
    if $x < 0$ **then**
        $result \leftarrow Negative\ input\ is\ not\ allowed.$
    else if $x > 110$ **then**
        $result \leftarrow Infinity$
    else
        $result \leftarrow 0, intervalGap \leftarrow 10^{-3}, i \leftarrow 0$
        while $i < a\ particular\ value$ **do**
            $result \leftarrow result + \frac{1}{2} * intervalGap * (f_y(i) + y(i - intervalGap))$
            $i \leftarrow i + intervalGap$
        end while
        $output \leftarrow result$
    end if
    return $output$
end function

function $pow(base, power)$
    $temPower \leftarrow power * logarithm(base)$
    $sum \leftarrow 1.0, flag \leftarrow 0$
    if $tempPower < 0$ **then**
        $tempPower \leftarrow temPower * -1$
        $flag \leftarrow 1$
    end if
    $ratio \leftarrow tempPower$
    for $i\ in\ (2, 500)$ **do**
        $ratio \leftarrow ratio * temPower/i$
        $sum \leftarrow sum + ratio$
    end for
    if $flag == 1$ **then**
        $sum \leftarrow 1.0/sum$
    end if
    return sum
end function

function $logarithmFunction(base)$
    $sum \leftarrow 0$
    $multiple \leftarrow 1.0$
    for $i\ in\ (1, 10000)$ **do**
        $multiple = multiple * ((base - 1)/(base + 1))$
        if $i \mod 2! = 0$ **then**
            $sum = sum + multiple/i$
        end if
    end for
    return $2 * sum$
end function

Algorithm 2 Calculate Gamma function using Lanczos approximation

Input: Array p as a coefficients, and a constant EPSILON
Output: Output value of Gamma function

function $gamma(x)$
  if $x < 0$ **then**
    return *Negative input is not allowed.*
  end if
  if $x < 0.5$ **then**
    return $\frac{\pi}{\sin(\pi * z) * gamma(1-z)}$
  else
    $x \leftarrow x - 1$
    $z \leftarrow 0.99999999999980993$
    for $(i, pval)\ in\ p$ **do**
      $z \leftarrow z + \frac{pval}{x+i+1}$
    end for
    $t \leftarrow x + length\,(p) - 0.5$
    $m \leftarrow \sqrt{2 * \pi} * pow\,(t, (x + 0.5)) * exp\,(-t) * z$
  end if
  return m
end function

function $sin(x)$
  $sum \leftarrow 0$
  $flag \leftarrow 1$
  for $i\ in\ (1, 500)$ **do**
    if $i\ mod\ 2 == 1$ **then**
      $N \leftarrow 0$
      $D \leftarrow 0$
      for $j\ in\ (1, i)$ **do**
        $N \leftarrow N * x$
        $D \leftarrow D * j$
      end for
      $sum \leftarrow sum + flag * \frac{N}{D}$
      $flag \leftarrow flag * -1$
    end if
  end for
end function

0.3.4 Advantages and Disadvantages

Algorithm 1

Advantages:

1. Using the the integral method makes the algorithm more intuitive and easy to understand.
2. More precise calculations results can be obtained.
3. The entire algorithm involves loops without recurtion, it requires less memory from computing devices.

Disadvantages:

1. When the input value is large, it takes significantly longer to produce the result.

Algorithm 2

Advantages:

1. The algorithm makes computing the gamma function becomes a matter of evaluating only a small number of elementary functons and multiplying by stored constants. Simpler to implement.
2. Calculation time is almost independent of the input value.

Disadvantages:

1. Less precise calculations results than Algorithm 1.
2. The coefficients and Constants that we give in advance are not always precise and accurate.

0.3.5 Final Decision

The algorithm 1 is chosen for implementation. Because it could provide more precise results, although it is theoretically slower than the second one, there is no much difference in terms of actual execution time. Figure 3 below shows the mind map.

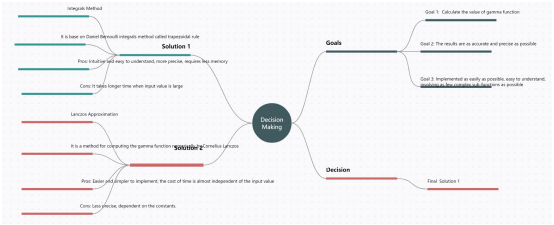
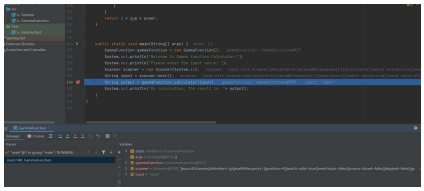


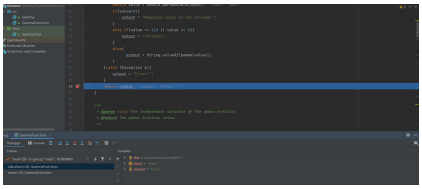
Figure 3: Mind map

0.4 Implementation

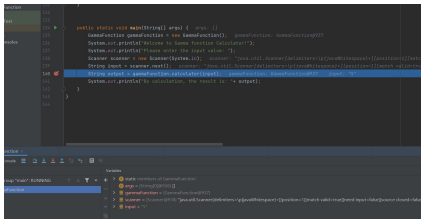
0.4.1 Evidence of Debugger



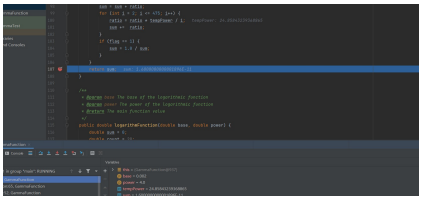
(a) debug_1



(b) debug_2



(c) debug_3



(d) debug_4

Figure 4: This is the Screenshot for debuggers

0.4.2 Evidence of Use of Pragmatic Quality Checking Tool

For Pragmatic Quality Checking, I used QAPLug, an IDE plugin that automatically evaluates code and projects in terms of performance, maintainability, executability, reliability, and usability, and the results are shown in the Figure 5 below.

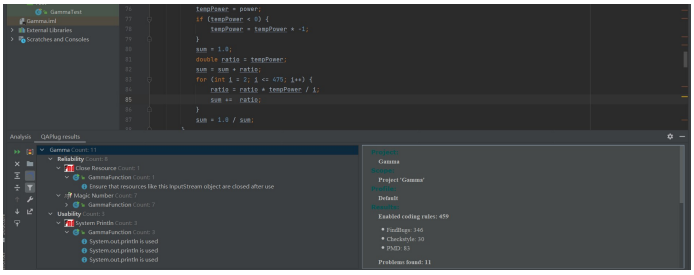


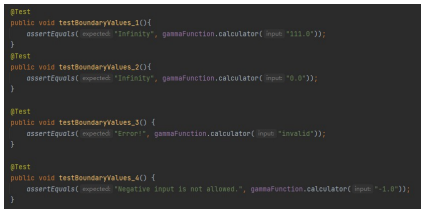
Figure 5: QA

0.5 Test

The five guidelines for unit tests proposed by A. Starreveld. Medium[8] are: (1) Unit tests have one assertion per test; (2) Avoid if-statements in a test; (3) Unit tests only "new()" the unit under test; (4) Unit tests do not contain hard-code values unless there is a special meaning; (5) Unit tests are stateless. I apply these five guidelines to our unit tests, but don't follow them exactly because Junit provides some other ways to avoid "bad" situations in testing.
First I used the Unit test framework, Junit test provides the form of annotations to facilitate the management of the test process, such as @Before annotation, the method under this annotation will be executed before the test case, so I instantiate the object written under this method. The test only needs a GammaFunction instance, no need to create instances for other classes, to a certain extent Follow the guidance (3) to avoid not finding the corresponding instance and unnecessary extra tests when tracing the cause of test case failure. As shown in Figure 6:

Figure 6: init test

When testing gamma functions for boundary and special values, I follow the test guide (1) and include an assertion in each test function. For the previous requirements, all test results meet the requirements. In particular, at a test input value of 0.5, the output results have an error range greater than 0.001 from the standard results, but also meet the requirement for that error to be less than 0.1. As shown in Figure 7:



(a) test2

(b) test3

Figure 7: Test for boundary and special values

Finally, I also tested all the sub-functions associated with the gamma function in Algorithm 2, and all the functions were calculated with an error of less than 0.001, as required. As shown in Figure 7:

Figure 8: sub-functions test

Bibliography

[1] Wikipedia contributors. Gamma function — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[2] Wikipedia contributors. Particular values of the gamma function — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[3] Wikipedia contributors. Trapezoidal rule — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[4] Wikipedia contributors. Lanczos approximation — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[5] Wikipedia contributors. Stirling's approximation — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[6] Wikipedia contributors. Pseudocode — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[7] Wikipedia contributors. Taylor series — Wikipedia, the free encyclopedia, 2022. [Online; accessed 5-August-2022].

[8] Albert Starreveld. The 5 unit testing guidelines. [Online; accessed 30-May-2019].