BETA FUNCTION CALCULATION

Indraneel Rachakonda, 40206072

Introduction

Beta function is also known as Euler's integral of first kind and is very important in calculus and is basically an association between input and output values. The beta function is used to determine average time to complete some tasks in the time related problems. Beta function has a very close connection to the gamma function which also the generalisation of the factorial function. The beta function is defined as follows:

$$\beta(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt$$

for Re (x) $>$ 0 and Re (y) $>$ 0

Domain

Re (x) $>$ 0 and Re (y) $>$ 0

Co-domain

All real numbers

$$(-\infty, \infty)$$

Characteristics

- The beta function is uniquely defined for positive numbers and complex numbers with positive real parts. It is approximated for other numbers.
- Beta function is symmetric,
$$\beta(x, y) = \beta(y, x)$$
- A key property of beta function is that it is closly related to the gamma function
$$\beta(x, y) = \frac{\gamma(x)\gamma(y)}{\gamma(x+y)}$$

Applications

- The Beta function was the first known 'Scattering' amplitude in String theory.
- In physics and string theory the beta function is used to calculate and reproduce scattering amplitudes in terms of the Regge trajectories.

Functional Requirements

- **R-1**: The Beta Function can provide an output if it has an input from a domain comprising of positive Real Numbers only.
- **R-2**: The Beta Function , can only provide an output if its parameters , X and Y are both positive numbers i.e. Real Number X $>$ 0 and Real Number Y $>$ 0.
- **R-3**:To compute the value of the Beta Function , a subordinate function needs to be used to calculate the value of A raised to the power B. In other words we need to define a power function to calculate A^B .

Non Functional Requirements

- **R-1**: The method used to calculate the Beta Function , should be scalable for different input values and different Hardware Requirements.

Assumptions

There are following assumptions made for the class project for the Beta function:

- **A-1**: The function will only accept the real positive numbers to limit the scope of the function.

Following is the pseudo-code for the Beta function:

Algorithms:

Gamma function and Beta function relation:

My first decision is to use the Gamma function for calculating the Beta function, as the implementation becomes much easier. As I have stated in the project-description, the Beta function is closely related to the Gamma function as;

$$\beta(x, y) = \frac{\gamma(x)\gamma(y)}{\gamma(x+y)}$$

So, by this it is clear that gamma function helps in calculating the beta function as well.

Algorithms for Gamma function:

There are two different algorithms or approximation techniques by which the Gamma function can be calculated. They are *Stirling's Approximation* and *Lanczos approximation*, and for the ease of my project I have decided to go with *Stirling's Approximation*. The reasons for this choice are explained below;

Algorithm 1 Calculate the Beta function

```
1:  procedure BETA(x, y)
2:      System Initialization
3:      Read input values
4:      if (x and y > 0) then
5:          betaNum ← GAMMA(x) * GAMMA(y);
6:          betaDen ← GAMMA(x + y);
7:          result ← betaNum÷ betaDen
8:          return result;
9:      else
10:         return ;
11:
12: procedure GAMMA(num)
13:     pi ← 3.1416;                    ▷ Pi value is used in gamma function
14:     eBase ← 2.7183;                 ▷ This is the base of natural logarithm
15:     varA ← (2 × pi) ÷ num;
16:     varB ← num ÷ eBase;
17:     squareRoot ← √varA;
18:     powerUp ← varB^num;
19:     result ← squareRoot × powerUp;
20:     return result;
```

- **Stirling's Approximation**: This is basically the approximation for *factorials* which is also very accurate when we are calculating the Gamma function for very small numbers as well, eg. 0.0001, which is one of the reasons to choose this approximation technique. Another reason is that the Big O notation of the *Stirling's Approximation* is O($\log_2$ n)
- **Lanczos Approximation**: This is basically for calculating the Gamma function numerically. It focuses more on precision on decimal places and is relatively complex than the *Stirling's Approximation*.

DVCS

Following is the address of the GitHub,
https://github.com/Indraneel-concordia/SOEN6011-Project

Debugger

I am using the inbuilt debugger from the Eclipse Oxygen IDE for debugging my code throughout and also testing my methods for accuracy and testing purpose. Eclipse debugger is the essential option to use as it is free and open source. Also since I have been using eclipse IDE for the better part of my student career, the familiarity to the eclipse debugger is another important reason for me to choose it.

Advantages

- It is free and open source.
- It supports other languages as well other than JAVA.
- Functionalities like step into code/ step out of code and step over code are provided.
- There is option for framework integration like JUnit etc.

Disadvantages

- It is usually slower than other debuggers like Visual studio.
- The independent processes cannot be attached to the eclipse debugger to debug and test.

Correctness

This source code calculates Beta function B(X, Y), for $x, y \subset (0, +\infty)$. The theory behind this program is based on mathematics integral calculation. The answer is obtained through solving mathematics integral by using java code implementation. This calculation does not import any packages, nor any build-in java functions. It combines empirical equation into Beta function:

$B(x, y) = \Gamma(x)\Gamma(y)/\Gamma(x + y)$

$B(x, y) = \int_0^\infty t^{x-1} \times (1-t)^{y-1} dt$

Efficiency

This program works with double datatype. This is a primitive datatype and it does not require much system memories. Generally speaking, for all values satisfying within the valid input range, and working on an average speed computer in 21 centuries, the program can generate results in milliseconds. The accuracy also decides the code efficiency. This program can estimate the solution in the accuracy of 15 digits after decimal point.

Maintainability and Program Style

Naming convention has applied. This provides a strong foundation for future maintenance. It is also easier for other developers to follow the current progress. Besides, documentation is necessary for engineers to trace back the developing history, which is an important resource to set up current maintainable decisions. This Beta Calculator program generates Javadoc, such as public class declaration and methods explanation have been added. Indentation and statement has been regulated. The code follows line wrapping and line break strictly. There is no extra whitespace. Exactly each section has one blank line to separate. For annotations, override, and static members have been used carefully with quality. Thus, by following the standard programming style, readability and maintainability have increases. Software engineers are easy to understand and catch up with other people's work.

Robustness and JUnit Testing

In order to increase the robustness, this program introduces Java JUnit testing. This program contains Junit assert .java file (*AssertTests.java*). In order to build a robust software. It demonstrates requirements and has covered in detailed Junit Assert test cases. Software testing is believed to be useful to increase software robustness. The program has passed all test cases.

Feasibility, Error handling, Usability

This program has a clear user interface. It is straight forward for user to understand the logic and instructions. This program also kindly reminds user if he has input a valid value. If error occurs, error handling exceptions shall be invoked. Try and catch blocks keep the program functioning, and exceptions have thrown error messages to remind user. This program also paid attentions in feasibility for user in the design phase, and has built to pursuit user-friendly aspect.

SonarLint

SonarLint is an IDE extension that helps you detect and fix quality issues as you write code. Like a spell checker, SonarLint squiggles flaws so that they can be fixed before committing code.

References

[1] Wikiedia: Beta Function,
`https://en.wikipedia.org/wiki/Beta_function`

[2] Wikiedia: Stirling's approximation,
`https://en.wikipedia.org/wiki/Stirling%27s_approximation`

[3] Wikiedia: Lanczos approximation,
`https://en.wikipedia.org/wiki/Lanczos_approximation`

[4] Beta function and its Applications. *by Riddhi D.* Department of Physics and Astronomy The University of Tennessee Knoxville, TN 37919, USA

[5] Wikiedia: Beta Function,
`https://en.wikipedia.org/wiki/Beta_function`

[6] Brilliant: The Beta Function,
`https://brilliant.org/wiki/beta-function/`

[7] Quora: What is a Beta Function,
`https://www.quora.com/What-is-beta-function`

[8] Google Java Style Guide. Retrieved July 26, 2019,
$ https://google.github.io/styleguide/javaguide.html