Assembly Language and Computer Architecture Lab

**MINI PROJECT REPORT**

Student: Chu Thạch Thảo

Student ID: 20194849

1. **Naïve method overview**

- Naïve method we’re considering here is actually Trapezoidal Rule. To calculate the integration the total area is divided into small trapezoids. Sum of area of all the trapezoids is the answer of our problem.

- It is easy to understand to human and also easy to implement in the computer. The precision of this method depends on the number of trapezoids we divide. More trapezoids means higher precision.

- This rule is used for approximating the difinite integrals where it uses the linear approximations of the functions.

- Here we’ll experience naïve method in three ways: even division (height of all trapezoids are the same) integrate from left to right and right to left, sloppily small (height of trapezoids increase from left to right) integrate from right to left.

1. **Result**

* Case 1: a = 0, b = 2, n = 10

integral-calculator: 4.428594871176362

Even division - left to right: 4.426463558338128

Even division - right to left: 4.42646352627467

Sloppily small - right to left: 4.549573503465402

* Case 2: a = 0, b = 4, n = 20

integral-calculator: 5.30327065467213

Even division - left to right: 5.302901700779358

Even division - right to left: 5.302901924703356

Sloppily small - right to left: 5.584574132449071

* Case 3: a = 0, b = 100, n = 10000

integral-calculator: 6.243186640432926

Even division - left to right: 6.24318662919517

Even division - right to left: 6.243186904586415

Sloppily small - right to left: NaN

* Case 4: a = 0, b = 1

integral-calculator: 3.999999999999987 x 10^(−7)

* + n = 10

Even division - left to right: 3.139926017069971

Even division - right to left: 3.1399259908390924

Sloppily small - right to left: 3.1492665141208205

* + n = 1000

Even division - left to right: 3.141592492903594

Even division - right to left: 3.141592481345689

Sloppily small - right to left: NaN

* + n = 10000

Even division - left to right: 3.141592651694072

Even division - right to left: 3.1415926502744327

Sloppily small - right to left: NaN

* Case 5: a = 0, b = 1e9

integral-calculator: 6.283185303179586

* + n = 1e5

Even division - left to right: 20000.000457950562

Even division - right to left: 2364736.0000033793

Sloppily small - right to left: NaN

* + n = 1e6

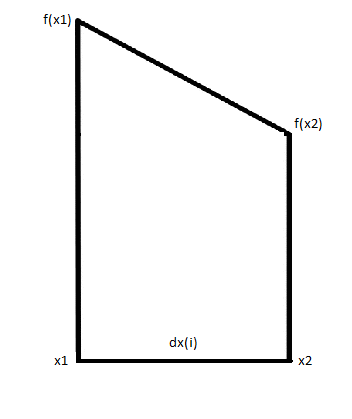
Even division - left to right: 2000.00648706424

Even division - right to left: -1.52127519538686E7

Sloppily small - right to left: NaN

1. **Implementation explain**

The implementation of this method is pretty straight-forward. Consider the bounds are a and b (a < b) and we divide the region into n parts.

A trapezoid i is like this:

The first trapezoid has x1 = a.

The nth trapezoid has x2 = b.

Finding x1 and x2 for each trapezoid is handled by a loop different for each way.

For even division dx = (b-a)/n = dx(i) with all i = 1…n

For sloppily small dx = (b-a)/(2^n-1) and dx(i) = 2^(i-1)dx

Finding f(x) is handled by procedure calculate\_f

Finding the area of the trapezoid is handled by procedure area

The final answer is sum of all area of the trapezoids (1) to (n)

1. **Things could go wrong**

The precision of this method is not high because there are many steps that errors can occur.

The first problem is the number of trapezoids we divide n. Mathematically, larger n results in more precise answer. However, it’s difficult to calculate with big n in computers because the error will also be big and the runtime is not efficient.

Second is the step of finding dx. For sloppily small the error is huge because with big n we have 2^n is very big and calculating this in computer (shift 1 bit to the left) we’ll lose a lot of bits, and the division is 99% not correct because b-a << 2^n-1. For even division the error is smaller but can still occur.

Third problem is calculating f(x). We have f(x) = 4/(x^2+1). With big x we’ll have big x^2 and normally we’ll see that 4 << x^2+1 and the division will have error.

Errors in other steps of the program will pile up from the problem of dx because we use dx in calculating the area of the trapezoid. And since we have many components in the sum, larger n means larger error.

So sum up, in theory larger number of trapezoids will give more precise answer but that is not the case in implementing this method on computer.

- Pros: This implementation is straightforward according to the analyzing stated in part 1

- Cons: The loop is time consuming, the result has unacceptable accuracy