Aproksimasi Perhitungan Integral Tentu Menggunakan Algoritma Divide and Conquer

Rahmah Khoirussyifa' Nurdini 13519013
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13519013@std.stei.itb.ac.id

***Abstrak*—Integral merupakan sebuah konsep matematika yang sering digunakan dalam ilmu Kalkulus. Integral dapat dihitung secara *indefinite* atau *definite* pada interval tertentu. Integral tentu dapat dihitung secara sederhana menggunakan berbagai pendekatan di mana area integrasi pada interval tersebut dibagi menjadi beberapa partisi dan pemjumlahan luas partisi tersebut dapat diaproksimasi sebagai hasil integrasi dari sebuah fungsi. Penjumlahan luas partisi tersebut biasanya dilakukan dengan algoritma *Brute Force* yang kurang efektif. Untuk itu, digunakan algoritma *Divide and Conquer* yang dapat memangkuskan proses penjumlahan luas partisi untuk aproksimasi perhitungan integral tentu ini.**

Kata Kunci—Integral tentu, aproksimasi, Divide and Conquer, Riemann Sum

I. PENDAHULUAN

Pada studi Matematika, terdapat cabang ilmu Kalkulus yang pada intinya mempelajari tentang suatu perubahan. Ilmu Kalkulus ini menyediakan sebuah *framework* yang membuat sistem pemodelan untuk sebuah perubahan. Selain itu, ilmu ini juga menyediakan cara untuk membuat prediksi dari perubahan dengan model-model tertentu. Ide dasar ilmu ini adalah untuk mempelajari perubahan menggunakan pendekatan perubahan yang instan atau perubahan yang terjadi dalam interval waktu yang sangat singkat.

Ilmu Kalkulus memiliki banyak cakupan konsep di dalamnya, seperti limit, turunan, integral, maupun deret. Pada makalah ini, penulis akan lebih membahas mengenai konsep integral. Integral sendiri merupakan konsep matematika yang dapat diinterpretasikan sebagai area atau daerah yang tercakupi dari sebuah kurva hasil fungsi satu peubah maupun banyak peubah. Integral juga biasa disebut dengan antiturunan.

Proses perhitungan integral ini biasa disebut dengan integrasi, sedangkan proses perhitungan integral dengan aproksimasi atau pendekatan biasa disebut dengan integral numerik. Terdapat banyak metode untuk melakukan perhitungan integral menggunakan aproksimasi. Beberapa di antaranya adalah *Riemann sum, Newton-Cotes formula, Simpson's rule, Boole's rule,* dan *Romberg integration*.

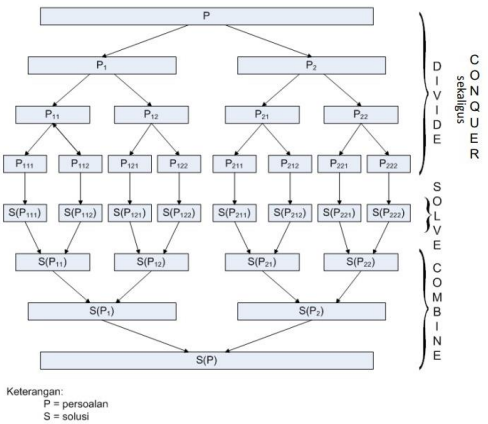
Pada makalah ini, penulis akan lebih membahas mengenai aproksimasi perhitungan integral tentu menggunakan *Riemann Sum* dan *Trapezoidal Rule* karena kedua metode ini sering dijumpai pada materi dasar ilmu Kalkulus. Kedua metode ini juga bisa dijadikan sebagai persoalan yang dapat dibagi menjadi beberapa upa-persoalan dengan karakteristik yang sama. Hal ini berarti persoalan tersebut dapat diselesaikan menggunakan algoritma *Divide and Conquer*. Penyelesaian menggunakan algoritma *Divide and Conquer* ini dirasa lebih baik karena memiliki kompleksitas waktu untuk kasus rata-rata yang lebih rendah dibandingkan menggunakan algoritma *Brute Force*.

II. LANDASAN TEORI

A. Algortima Divide and Conquer

1. Definisi

Algoritma *Divide and Conquer* ini berasal dari tiga kata, yaitu, *divide*, *conquer*, dan *combine*. *Divide* berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semua, tetapi berukuran lebih kecil. *Conquer* berarti menyelesaikan masing-masing upa-persoalan yang dilakukan secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar. *Combine* berarti menggabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semua. Berikut ini adalah visualisasi dari cara kerja algoritma *Divide and Conquer*.



Gambar 2.1 Ilustrasi *Divide and Conquer*

(Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf)

2. *Skema Umum*

Secara umum, persoalan yang bisa diselesaikan dengan algoritma *Divide and Conquer* ini adalah persoalan yang dapat dibagi menjadi beberapa upa-persoalan dengan karakteristik yang sama dengan persoalan semula. Biasanya, penyelesaian persoalan menggunakan algoritma *Divide and Conquer* ini diungkapkan dalam skema rekursif. Berikut adalah gambaran algoritma penyelesaian menggunakan metode *Divide and Conquer* secara umum.

```
procedure DIVIDEandCONQUER(input P: problem, n: integer)
{ Menyelesaikan persoalan P dengan Algoritma Divide and Conquer. Masukan: masukan persoalan P berukuran n Luaran: solusi dari persoalan semula }

Deklarasi
r : integer

Algoritma
{ Jika ukuran persoalan P sudah cukup kecil }
if n ≤ n0 then
   SOLVE persoalan P yang berukuran n ini
else
   DIVIDE menjadi r upa-persoalan, P1, P2, …, Pr, yang masing-masing berukuran n1, n2, …, nr

   for masing-masing P1, P2, …, Pr, do
      DIVIDEandCONQUER(Pi, ni)
   endfor

   COMBINE solusi dari P1, P2, …, Pr menjadi solusi persoalan semula

endif
```

3. *Kompleksitas*

Secara umum, persoalan yang dapat diselesaikan dengan algoritma *Divide and Conquer* memiliki kompleksitas waktu yang dapat dihitung menggunakan persamaan berikut.

$$T(n) = \begin{cases} g(n) & ,n \le n_0 \\ T(n_1) + T(n_2) \dots + T(n_r) + f(n) & ,n > n_0 \end{cases} \quad (1)$$

dengan *T(n)* adalah kompleksitas waktu penyelesaian persoalan P yang berukuran *n*, *g(n)* adalah kompleksitas waktu untuk *SOLVE* jika *n* sudah berukuran kecil, $T(n_1) + T(n_2) + \dots + T(n_r)$ adalah kompleksitas waktu untuk memproses setiap upa-persoalan, dan *f(n)* adalah kompleksitas waktu untuk *COMBINE* solusi dari masing-masing upa-persoalan.

Secara rata-rata, algoritma ini memiliki kompleksitas waktu yang jauh lebih baik daripada algoritma *Brute Force* karena algortima ini membagi persoalan yang besar menjadi upa-persoalan yang lebih kecil sehingga penyelesaian upa-persoalan ini dapat dilakukan dengan lebih mangkus.

B. *Aproksimasi Integral Tentu*

1. *Definisi Aproksimasi Integral Tentu*

Integral tentu untuk fungsi satu peubah ini dapat dihitung secara sederhana dengan berbagai metode pendekatan atau aproksimasi. Perhitungan integral tentu untuk fungsi satu peubah yang membentuk sebuah kurva pada koordinat kartesian dua dimensi ini dapat dilakukan dengan membagi area yang ada di antara kurva dan sumbu *x* menjadi beberapa partisi dalam bentuk bangun datar sederhana, seperti bujur sangkar atau trapesium.

Jika area tersebut dibagi menjadi *N* partisi, maka nilai *x* untuk setiap partisi dapat ditentukan sebagai berikut. Nilai *a* atau *lower limit* akan menjadi x_0 dan nilai *b* atau *upper limit* akan menjadi x_N.

$$x_0 = a < x_1 < \cdots < x_{N-1} < x_N = b \quad (2)$$

Penjumlahan luas daerah setiap partisi dapat dinotasikan sebagai berikut.

$$\sum_{i=1}^{N} f(x_i^*)(x_i - x_{i-1}) \ , \ x_i^* \in [x_{i-1}, x_i] \quad (3)$$

dengan x_i^* dapat bernilai minimal sampai maksimal pada partisi tersebut, tergantung jenis pendekatan yang akan dipakai untuk perhitungan nilai integral tentu ini. Jika disimpulkan, perhitungan nilai integral tentu dapat didekati seperti notasi di bawah ini.

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} f(x_i^*)(x_i - x_{i-1}) \ , \ x_i^* \in [x_{i-1}, x_i] \quad (4)$$

2. *Beberapa Metode Aproksimasi*

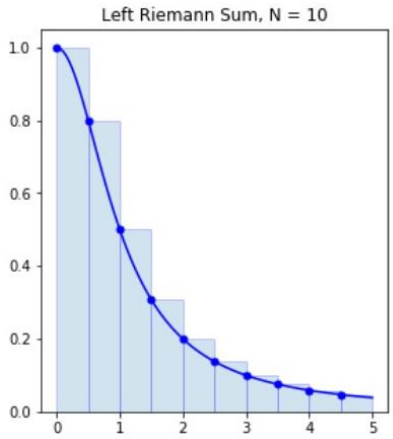
Terdapat beberapa metode aproksimasi untuk menghitung integral tentu suatu fungsi dengan satu peubah. Beberapa di antaranya adalah *Riemann Sum* (*Left Riemann Sum*, *Right Riemann Sum*, dan *Midpoint Riemann Sum*) dan *Trapezoidal Rule*. Berikut ini adalah penjelasan untuk masing-masing metode aproksimasi.

a. *Left Riemann Sum*

Dengan metode ini, digunakan pendekatan

bentuk partisi area di antara kurva dan sumbu *x* dalam bentuk persegi panjang dengan lebar partisi yang sama sesuai jumlah partisi dan panjang partisi yang sesuai dengan nilai *f(x)* pada *left endpoint* dari partisi tersebut. Pada persamaan (3) dan (4), x_i^* bernilai x_{i-1} untuk metode *Left Riemann Sum* ini.

Visualisasi dari *Left Riemann Sum* ini dapat dilihat pada gambar berikut.

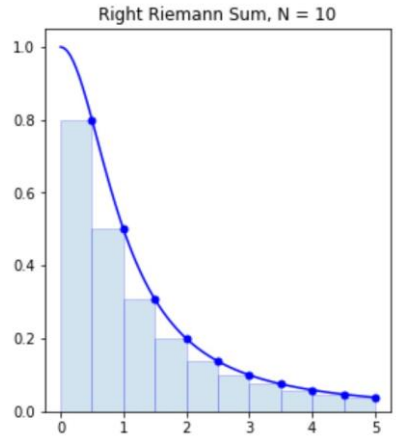


Gambar 2.2 Visualisasi *Left Riemann Sum*
(Sumber: https://www.math.ubc.ca/~pwalls/math-python/integration/riemann-sums/)

b. Right Riemann Sum

Dengan metode ini, digunakan pendekatan bentuk partisi area di antara kurva dan sumbu *x* dalam bentuk persegi panjang dengan lebar partisi yang sama sesuai jumlah partisi dan panjang partisi yang sesuai dengan nilai *f(x)* pada *right endpoint* dari partisi tersebut. Pada persamaan (3) dan (4), x_i^* bernilai x_i untuk metode *Right Riemann Sum* ini.

Visualisasi dari *Right Riemann Sum* ini dapat dilihat pada gambar berikut.

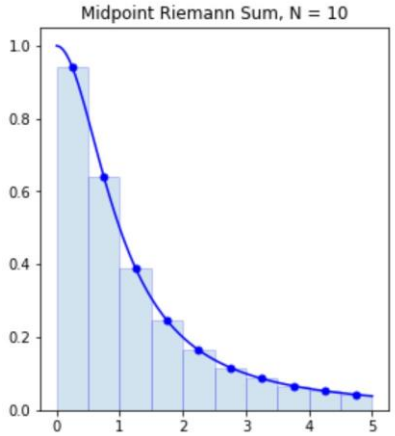


Gambar 2.3 Visualisasi *Right Riemann Sum*
(Sumber: https://www.math.ubc.ca/~pwalls/math-python/integration/riemann-sums/)

c. Midpoint Riemann Sum

Dengan metode ini, digunakan pendekatan bentuk partisi area di antara kurva dan sumbu *x* dalam bentuk persegi panjang dengan lebar partisi yang sama sesuai jumlah partisi dan panjang partisi yang sesuai dengan nilai tengah *f(x)* atau kurva dari partisi tersebut. Pada persamaan (3) dan (4), x_i^* bernilai $(x_{i-1}+x_i)/2$ untuk metode *Midpoint Riemann Sum* ini.

Visualisasi dari *Midpoint Riemann Sum* ini dapat dilihat pada gambar berikut.



Gambar 2.4 Visualisasi *Midpoint Riemann Sum*
(Sumber: https://www.math.ubc.ca/~pwalls/math-python/integration/riemann-sums/)

d. Trapezoidal Rule

Dengan metode ini, digunakan pendekatan bentuk partisi area di bawah kurva dalam bentuk trapesium dengan panjang sisi ‘atas’ trapesium adalah nilai *f(x)* pada *left endpoint* dari partisi, panjang sisi ‘bawah’ trapesium adalah nilai *f(x)* pada *right endpoint* dari partisi, dan tinggi trapesium adalah lebar partisi sesuai dengan jumlah partisi yang ditentukan. Sebagai catatan, sisi ‘atas’ dan ‘bawah’ mungkin ditukar nilainya.

Berbeda dengan ketiga metode sebelumnya, metode *Trapezoidal Rule* ini dapat dihitung menggunakan persamaan berikut ini.

$$T_N(f) = \frac{\Delta x}{2} \sum_{i=1}^{N} (f(x_i) + f(x_{i-1})) \quad (5)$$

dengan $\Delta x = (b-a)/N$ yang merupakan lebar partisi dan $x_i = i * \Delta x$.

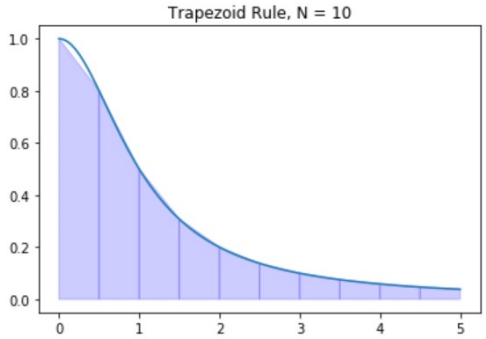
Jika diperhatikan, hasil aproksimasi menggunakan *Trapezoidal Rule* ini merupakan nilai rata-rata dari hasil aproksimasi

menggunakan *Left Riemann Sum* dan *Right Riemann Sum*. Hal ini dapat dilihat dari persamaan berikut.

$$T_N(f) = \frac{\Delta x}{2}\sum_{i=1}^{N}(f(x_i) + f(x_{i-1}))$$

$$= \frac{1}{2}\left(\sum_{i=1}^{N} f(x_i)\Delta x + \sum_{i=1}^{N} f(x_{i-1})\Delta x\right) \quad (6)$$

Visualisasi dari *Trapezoidal Rule* ini dapat dilihat pada gambar berikut.



Gambar 2.5 Visualisasi *Trapezoidal Rule*
(Sumber: https://www.math.ubc.ca/~pwalls/math-python/integration/trapezoid-rule/)

III. PEMBAHASAN

A. *Pemetaan Masalah Integral Tentu ke Algoritma Divide and Conquer*

Persoalan aproksimasi perhitungan integral tentu ini dapat dibagi menjadi beberapa upa-persoalan dengan karakteristik yang serupa. Perhitungan integral tentu ini dilakukan dengan membagi area antara kurva dan sumbu *x* sebanyak *N* partisi. Semakin besar nilai *N*, semakin baik hasil aproksimasinya karena perhitungan integral tentu ini akan semakin akurat.

Jika sudah ditentukan jumlah partisi yang akan membagi area integrasi tersebut, lebar setiap partisi tersebut (Δx) dapat ditentukan dengan menghitung *upper limit* dikurangi *lower limit* dibagi dengan jumlah partisi. Lebar partisi inilah yang nanti akan menentukan apakah sebuah upa-persoalan sudah cukup kecil untuk diselesaikan dengan algoritma *Divide and Conquer*.

Untuk menyelesaikan persoalan ini, pertama-tama, definisikan terlebih dahulu fungsi satu peubah yang akan dihitung nilai integral tentunya. Kemudian, tentukan batas bawah/*lower limit* (*a*) dan batas atas/*upper limit* (*b*) dari integral tentu yang ingin didekati perhitungannya. Lalu, tentukan jumlah partisi dan hitung lebar partisi dari area integrasi yang ingin dilakukan.

Setelah itu, dengan algoritma *Divide and Conquer*, dibuat fungsi rekursif yang akan mengecek apakah lebar dari sebuah partisi pada upa-persoalan tersebut (dapat dihitung dengan *b*-*a*) sudah cukup kecil atau bernilai kurang sama dengan lebar partisi yang ditentukan dari awal. Jika sudah, maka fungsi akan mengembalikan penyelesaian dari upa-persoalan ini (*SOLVE*), yaitu luas dari partisi yang sudah cukup kecil tersebut, tergantung tipe aproksimasi yang ditentukan dari awal.

Jika lebar partisi pada upa-persoalan tersebut masih cukup besar, maka bagi upa-persoalan (*DIVIDE*) tersebut menjadi dua bagian yang cenderung sama besar, yaitu P_1 dan P_2. P_1 dan P_2 ini kemudian akan diselesaikan dengan fungsi rekursif yang sama. Setelah mendapat penyelesaian dari P_1 dan P_2, kita kombinasikan (*COMBINE*) solusi dari kedua upa-persoalan tersebut dan fungsi rekursif akan mengembalikan hasil kombinasi solusinya.

Detail dari algoritma ini dapat dilihat pada implementasi fungsi rekursif dalam notasi algoritmik pada bagian *Implementasi dalam Notasi Algoritmik*.

B. *Implementasi dalam Notasi Algoritmik*

```
function calculate(type: string, fx:
function, a, b, width: real, N: integer) →
real
{Menghitung seluruh luas partisi dari fungsi
fx dengan interval dari a sampai b dengan N
buah partisi yang memiliki lebar sebesar
width. Fungsi dibuat secara rekursif}

KAMUS
 N_1, N_2: integer {Jumlah partisi dari P_1
           dan P_2}
 a_1, b_1: real {Batas bawah (a) dan atas
           (b) dari P_1}
 a_2, b_2: real {Batas bawah (a) dan atas
           (b) dari P_2}

ALGORITMA
 { Jika persoalan sudah cukup kecil }
 if (b-a) <= width then
    { SOLVE persoalan sesuai tipenya }
    depend on (type):
    type = "left" : → fx(a) * width
    type = "mid" : → fx((a+b)/2) * width
    type = "right" : → fx(b) * width
    type = "trapezoid" : →
    (fx(a)+fx(b))*width*0.5

 { Jika persoalan masih cukup besar }
 else
    N_1 ← N div 2 {DIVIDE persoalan jadi 2}
    N_2 ← N - N_1

    a_1 ← a
    b_1 ← a + N_1 * width

    a_2 ← b_1
    b_2 ← b

 { COMBINE solusi dari P_1 + P_2 }
 → calculate(type, fx, a_1, b_1, N_1, width)
   + calculate(type, fx, a_2, b_2, N_2,
   width
```

Selain pada notasi algoritmik di atas, penulis juga membuat kode program untuk memvisualisasikan dan memberikan solusi atas beberapa kasus uji yang ada di bagian *Kasus Uji dan Performa* pada link: https://github.com/rahmahkn/IF2211-Paper-DivideAndConquer.

C. *Kasus Uji dan Performa*

1. *Kasus Uji 1*

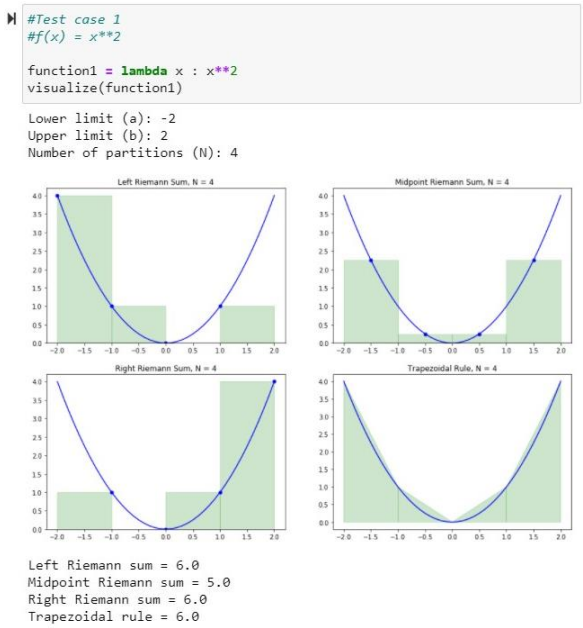
Pada kasus ini, diuji aproksimasi perhitungan integral tentu sebagai berikut.

$$\int_{-2}^{2} x^2 dx$$

Hasil asli dari perhitungan integral di atas adalah sebagai berikut.

$$\int_{-2}^{2} x^2\,dx = \frac{1}{3}x^3\Big|_{-2}^{2} = \frac{16}{3}$$

Sedangkan untuk aproksimasi perhitungan integral ini digunakan 4 buah partisi dengan lebar yang sama. Berikut adalah hasil visualisasi dan aproksimasi perhitungan nilai integral di atas.



Gambar 3.1 Hasil Kasus Uji 1
(Sumber: Dokumentasi Pribadi)

Dari kasus uji ini, metode yang memberikan hasil dengan galat terkecil adalah *Midpoint Riemann Sum*.

2. *Kasus Uji 2*

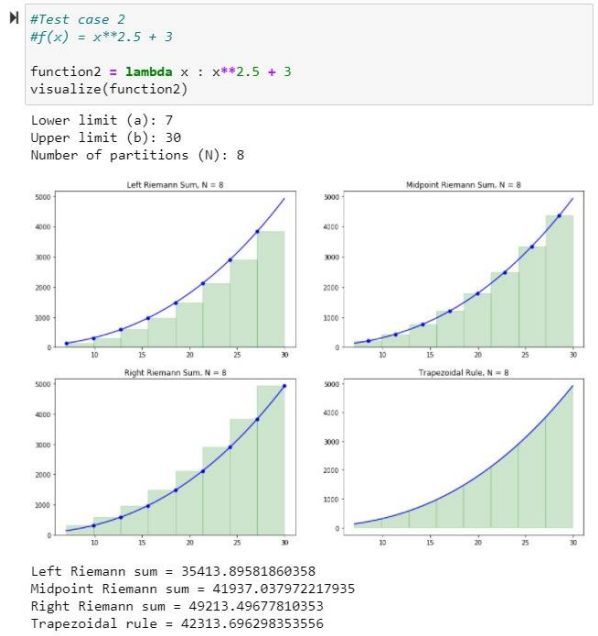
Pada kasus ini, diuji aproksimasi perhitungan integral tentu sebagai berikut.

$$\int_{7}^{30} x^2\sqrt{x} + 3\,dx$$

Hasil asli dari perhitungan integral di atas adalah sebagai berikut.

$$\int_{7}^{30} x^2\sqrt{x} + 3\,dx = \frac{2}{7}x^{\frac{7}{2}} + 3x\Big|_{7}^{30} = 42062.5994$$

Sedangkan untuk aproksimasi perhitungan integral ini digunakan 8 buah partisi dengan lebar yang sama. Berikut adalah hasil visualisasi dan aproksimasi perhitungan nilai integral di atas.



Gambar 3.2 Hasil Kasus Uji 2
(Sumber: Dokumentasi Pribadi)

Dari kasus uji ini, metode yang memberikan hasil dengan galat terkecil adalah *Midpoint Riemann Sum*.

3. *Kasus Uji 3*

Pada kasus ini, diuji aproksimasi perhitungan integral tentu sebagai berikut.

$$\int_{10}^{100} \sqrt{x} - 2\,dx$$

Hasil asli dari perhitungan integral di atas adalah sebagai berikut.

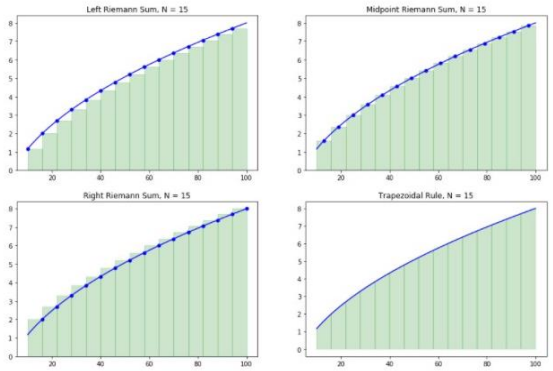
$$\int_{10}^{100} \sqrt{x} - 2\,dx = \frac{2}{5}x^{\frac{5}{2}} - 2x\Big|_{7}^{30} = 465.5848$$

Sedangkan untuk aproksimasi perhitungan integral ini digunakan 15 buah partisi dengan lebar yang sama. Berikut adalah hasil visualisasi dan aproksimasi perhitungan nilai integral di atas.

```
#Test case 3
#f(x) = x**0.5 - 2

function3 = lambda x : x**0.5 - 2
visualize(function3)
```

```
Lower limit (a): 10
Upper limit (b): 100
Number of partitions (N): 15
```




```
Left Riemann sum = 444.74930053355547
Midpoint Riemann sum = 465.7452540432074
Right Riemann sum = 485.7756345725452
Trapezoidal rule = 465.2624675530503
```

Gambar 3.3 Hasil Kasus Uji 3
(Sumber: Dokumentasi Pribadi)

Dari kasus uji ini, metode yang memberikan hasil dengan galat terkecil adalah *Midpoint Riemann Sum*.

4. *Kasus Uji 4*

Pada kasus ini, diuji aproksimasi perhitungan integral tentu sebagai berikut.

$$\int_{-5}^{7} x + 1 \, dx$$

Hasil asli dari perhitungan integral di atas adalah sebagai berikut.

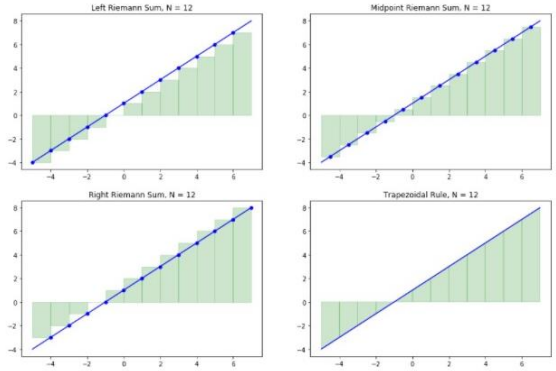
$$\int_{-5}^{7} x + 1 \, dx = \frac{1}{2}x^2 + x \Big|_{-5}^{7} = 24$$

Sedangkan untuk aproksimasi perhitungan integral ini digunakan 12 buah partisi dengan lebar yang sama. Berikut adalah hasil visualisasi dan aproksimasi perhitungan nilai integral di atas.

```
#Test case 4
#f(x) = x + 1

function4 = lambda x : x + 1
visualize(function4)
```

```
Lower limit (a): -5
Upper limit (b): 7
Number of partitions (N): 12
```




```
Left Riemann sum = 18.0
Midpoint Riemann sum = 24.0
Right Riemann sum = 30.0
Trapezoidal rule = 24.0
```

Gambar 3.4 Hasil Kasus Uji 4
(Sumber: Dokumentasi Pribadi)

Dari kasus uji ini, metode yang memberikan hasil dengan tepat adalah *Midpoint Riemann Sum* dan *Trapezoidal Rule*.

5. *Performa*

Untuk menguji apakah algoritma *Divide and Conquer* yang diterapkan untuk melakukan aproksimasi perhitungan ini hasilnya sudah sesuai, penulis menggunakan *tools* yang ada pada https://www.emathhelp.net/calculators/calculus-2/riemann-sum-calculator/.

Dari keempat kasus yang penulis ujikan, seluruh solusi yang dihasilkan sesuai dengan hasil perhitungan oleh *tools* pada *link* di atas. Dari keempat kasus uji ini pula, terbukti bahwa hasil aproksimasi menggunakan *Trapezoidal Rule* merupakan rata-rata dari hasil aproksimasi menggunakan *Left Riemann Sum* dan *Right Riemann Sum*.

IV. KESIMPULAN

Algoritma *Divide and Conquer* merupakan algoritma pencarian solusi dengan membagi persoalan besar menjadi beberapa upa-persoalan yang lebih kecil yang memiliki karakteristik yang sama. Jika dibandingkan dengan algoritma *Brute Force*, algoritma ini secara rata-rata dapat menyelesaikan berbagai macam persoalan dengan lebih mangkus. Salah satu persoalan yang dapat diselesaikan dengan algoritma ini adalah persoalan aproksimasi perhitungan integral tentu untuk fungsi satu peubah. Dari beberapa kasus yang diuji oleh penulis, terbukti bahwa algoritma *Divide and Conquer* dapat memberikan solusi yang tepat untuk setiap persoalan. Dari beberapa metode yang dipakai penulis, dapat disimpulkan

bahwa metode *Midpoint Riemann Sum* dan *Trapezoidal Rule* lebih akurat dalam memberikan solusi persoalan jika dibandingkan dengan kedua metode lainnya.

VIDEO LINK PADA YOUTUBE

Video pembahasan makalah ini dapat diakses pada tautan https://youtu.be/z_PJUQV60BE.

UCAPAN TERIMA KASIH

Penulis memanjatkan puji serta syukur kepada Allah SWT atas kelancaran selama pengerjaan makalah untuk mata kuliah Strategi Algoritma ini. Penulis mengucapkan terima kasih kepada orangtua yang selalu memberikan dukungan kepada penulis. Penulis juga mengucapkan terima kasih kepada seluruh dosen pengajar mata kuliah Strategi Algoritma, terutama Bapak Dr. Ir. Rila Mandala, M.Eng. sebagai dosen pengampu K1 yang sudah menyampaikan ilmunya. Tak lupa, penulis berterima kasih kepada teman-teman yang telah banyak memberi dukungan dan semangat kepada penulis selama proses penulisan makalah ini.

REFERENSI

[1] Rinaldi Munir, "Algoritma *Divide and Conquer* (Bagian 1)", 2021 (diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf pada 7 Mei 2021 pukul 11.00 WIB)

[2] Patrick Walls, "Riemann Sum", 2020 (diakses dari https://www.math.ubc.ca/~pwalls/math-python/integration/riemann-sums/ pada 7 Mei 2021 pukul 21.00 WIB)

[3] Patrick Walls, "Trapezoidal Rule", 2020 (diakses dari https://www.math.ubc.ca/~pwalls/math-python/integration/trapezoid-rule/ pada 8 Mei 2021 pukul 14.00 WIB)

[4] Khan Academy, "Riemann sums review", 2021 (diakses dari https://www.khanacademy.org/math/ap-calculus-ab/ab-integration-new/ab-6-2/a/riemann-sums-review pada 8 Mei 2021 pukul 22.00 WIB)

[5] MIT Mathematics, "What Is Calculus and Why do we Study it?" (diakses dari http://www-math.mit.edu/~djk/calculus_beginners/chapter00/section02.html pada 8 Mei 2021 pukul 22.30 WIB)

[6] Stover, Christopher and Weisstein, Eric W., "Integral." From MathWorld--A Wolfram Web Resource (diakses dari https://mathworld.wolfram.com/Integral.html pada 8 Mei 2021 pukul 23.30 WIB)

[7] Weisstein, Eric W., "Numerical Integration." From MathWorld--A Wolfram Web Resource (diakses dari https://mathworld.wolfram.com/NumericalIntegration.html pada 9 Mei 2021 pukul 12.00 WIB)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bangka Tengah, 11 Mei 2021

Rahmah Khoirussyifa' Nurdini
13519013