Implementasi Metode Jumlah Riemann untuk Mendekati Luas Daerah di Bawah Kurva Suatu Fungsi Polinom dengan *Divide and Conquer*

Dewita Sonya Tarabunga - 13515021
Program Studi Tenik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung 40132, Indonesia
dewitast20@gmail.com

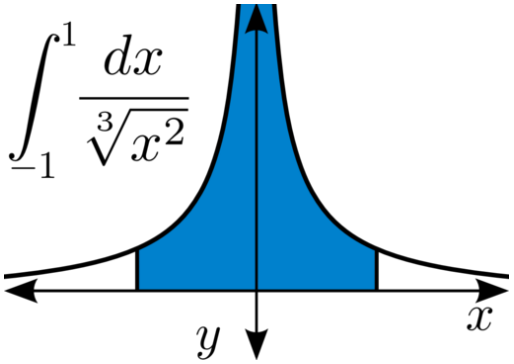
Abstract*—Permasalahan pencarian luas daerah di bawah kurva merupakan salah satu permasalahan yang sangat penting. Konsep integral juga merupakan konsep dasar yang sangat penting dan sangat banyak aplikasinya pada bidang-bidang lain, seperti pada rekayasa dan sebagainya. Namun, muncul suatu permasalahan, yaitu munculnya fungsi-fungsi aneh yang tidak dapat dihitung anti-turunannya sehingga perhitungan integral tentu nya harus dilakukan dengan pendekatan-pendekatan numerik. Pendekatan numerik pada matematika menerapkan konsep ketakterhinggaan, konsep yang tak dikenal komputer, sehingga perhitungan dengan komputer harus dilakukan dengan aproksimasi sampai batas tertentu. Pada makalah ini akan dibahas penerapan aproksimasi perhitungan luas daerah di bawah kurva dengan metode jumlah Riemann dengan algoritma *divide and conquer.

Keywords—divide and conquer; fungsi polinom; integral; jumlah Riemann; luas daerah

I. PENDAHULUAN

Dalam sejarah perkembangan bidang matematika, perhitungan luas daerah dan volume dari suatu bidang datar atau ruang sudah dipelajari sejak zaman dahulu kala. Salah satu contoh permasalahan yang berkembang pada zaman kuno adalah permasalahan menghitung luas lingkaran, yang dijawab oleh Archimedes dengan melakukan pendekatan luas lingkaran dengan menghitung luas segi-n beraturan. Semakin besar nilai n, maka akan semakin dekat luas segi-n beraturan tersebut dengan luas lingkaran. Metode tersebut dinamakan *methods of exhaustion.*

Permasalahan perhitungan luas dan volume, meskipun terlihat sederhana dan lebih ke geometri elementer, namun dalam aplikasi atau terapannya memiliki peran yang sangat penting dan beragam. Banyak masalah pada bidang fisika, biologi, maupun rekayasa yang memiliki titik pusat permasalahan pada luas atau volume benda. Misalnya menentukan titik pusat massa, dan lain sebagainya.



Gambar 1 Integral sebagai Luas Daerah di Bawah Kurva

Sumber : www.statisticshowto.com

Kalkulus dikembangkan oleh Isaac Newton dan Gottfried Leibniz pada abad ke-17. Seperti yang telah diketahui, mayoritas konsep-konsep kalkulus seperti limit menggunakan konsep ketakterhinggaan seperti *infinity* atau *infinitesimal. Infinitesimal* adalah suatu bilangan yang sangat kecil atau dapat dikatakan *infinitely close to* 0. Begitu pula dengan masalah penghitungan luas daerah di bawah kurva yang menggunakan pendekatan juga dengan konsep *infinitesimal.*

Secara historis, dalam bidang matematika, perkembangan integral muncul setelah sebelumnya Fermat dan Descartes telah mengembangkan konsep dari turunan dan konsep geometri analitik pada tahun 1630an. Tak lama setelah itu, tepatnya pada tahun 1660an, Isaac Newton memperkenalkan *inverse tangents* untuk menemukan luas area di bawah kurva. Namun, banyak fungsi-fungsi pada matematika lanjut yang sangat sulit untuk ditemukan integralnya, baik integral tentu maupun tak tentu, sehingga diperlukan cara lain untuk menghitung integral selain cara eksak, yaitu dengan menggunakan aproksimasi misalnya

$$\int_0^1 e^{x^2}\, dx$$

Yang tak dapat dihitung secara eksak dengan cara-cara elementer.

Pada perkembangan selanjutnya, sudah banyak metode-metode pada bidang numerik yang dikembangkan untuk menaksir luas daerah kurva. Jika digunakan pada pemrograman, metode-metode ini tentu sangat penting karena komputer tidak mengenal istilah tak hingga. Komputer memiliki keterbatasan dalam hal representasi bilangan-bilangan sehingga tidak mungkin mengimplementasikan suatu perhitungan yang benar-benar menggunakan konsep *infinite* atau *infinitesimal.* Namun, dengan menggunakan metode numerik atau *numeric analysis*, peneliti dapat menggunakan aproksimasi untuk menaksir hasil dari suatu komputasi dengan menggunakan komputer dengan "mendekati" tak hingga. Beberapa contoh jenis integral yang sudah berkembang adalah integral Riemann, integral Lebesgue, dan lain-lain. Namun, dalam makalah ini, penulis hanya akan membahas pendekatan perhitungan luas area di bawah kurva fungsi polinom dengan metode jumlah Riemann.

II. LANDASAN TEORI

A. *Integral Riemann*

Setelah pertama kali diperkenalkan oleh Isaac Newton dan Gottfried Leibniz pada 1660an, kalkulus semakin berkembang, termasuk juga teori-teori dari turunan maupun integral. Lalu, 2 abad kemudian, tepatnya pada tahun 1850an, seorang matematikawan bernama Bernhard Riemann memperkenalkan sudut pandang baru terhadap integral, yaitu memperkenalkan integral sebagai suatu jumlah luas area dan dipadu dengan konsep limit untuk mencari integral tentu dari suatu fungsi. Sebelum membahas definisi dari integral Riemann, akan diberikan beberapa definisi yang diperlukan untuk mendefinisikan integral Riemann.

Definisi 1. Suatu partisi $P \coloneqq (x_0, x_1, \dots, x_n)$ dari suatu interval tutup terbatas $I = [a, b]$ merupakan suatu himpunan terbatas yang terurut dimana

$$a = x_0 < x_1 < \cdots < x_n = b$$

Partisi *P* mempartisi interval *I* menjadi *n* subinterval yang saling tidak berpotongan $I_i, \forall i \in \{1,2, \dots, n\}$ dimana

$$I_i = [x_{i-1}, x_i], \qquad \forall i \in \{1,2, \dots, n\}$$

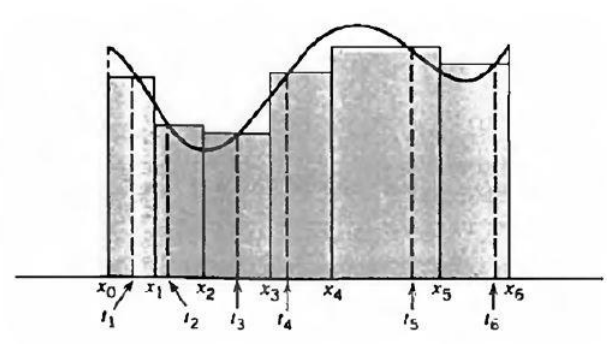
Definisi 2. Norma dari suatu partisi, dilambangkan dengan $||P||$ untuk suatu partisi $P \coloneqq (x_0, x_1, \dots, x_n)$ dari interval *I* merupakan jarak terbesar yang dimiliki dua elemen berurutan dari suatu partisi, atau secara formal,

$$\big||P|\big| = \max(x_1 - x_0, x_2 - x_1, \dots, x_n - x_{n-1})$$

Definisi 3. Suatu *tagged partition* adalah suatu partisi sesuai definisi 1 dimana di setiap interval I_i sudah dipilih suatu titik t_i dimana $t_i \in [x_{i-1}, x_i], \forall i \in \{1,2, \dots, n\}$. Titik t_i tersebut dinamakan *tag* dari suatu partisi I_i.

Dengan tiga definisi di atas, sudah dapat dilakukan pendefinisian dari jumlah Riemann. Tidak lain, jumlah Riemann dari suatu fungsi $f: [a, b] \to \mathbb{R}$ dengan suatu *tagged partition P* adalah suatu angka $S(f; P)$ dimana

$$S(f; P) \coloneqq \sum_{i=1}^{n} f(t_i)(x_i - x_{i-1})$$



Gambar 2 Ilustrasi dari jumlah Riemann

Sumber : Bartle, *Introduction to Real Analysis*

Dapat dilihat dari gambar bahwa jumlah Riemann berusaha untuk menaksir luas daerah di bawah kurva di gambar. Dapat dilihat juga bahwa jumlah Riemann tersebut memiliki kesalahan dalam perhitungan karena ada bagian bawah kurva yang tidak di *cover* oleh blok-blok partisi. Sebaliknya, terdapat pula daerah di atas kurva yang ter-*cover* oleh blok-blok partisi. Namun, secara intuitif kesalahan perhitungan karena aproksimasi tersebut dapat diminimalisir dengan memperkecil blok-blok partisi, dengan kata lain, memperkecil norma dari partisi juga akan memperkecil nilai kesalahan atau *error* dari jumlah Riemann dan membuat aproksimasi semakin baik.

Berikutnya, definisi integral Riemann berkembang dari intuisi di atas. Integral Riemann dari fungsi f pada interval $[a, b]$ terdefinisi jika f terintegralkan Riemann, yaitu jika terdapat bilangan $L \in \mathbb{R}$ sehingga $\forall \epsilon > 0$, $\exists \delta_\epsilon > 0$ sehingga jika *P* merupakan sembarang *tagged partition* dari $[a, b]$ dengan $\big||P|\big| < \delta_\varepsilon$, maka

$$|S(f; P) - L| < \epsilon$$

Lebih jauh, dapat dibuktikan bahwa bilangan *L* di atas unik dan merupakan limit dari jumlah Riemann, dan dikatakan nilai integral Riemann dari f, yaitu

$$L = \int_b^a f(x)dx$$

Dapat dilihat bahwa konsep di atas melibatkan *infinitesimal*, dan sudah dijelaskan pula sebelumnya bahwa komputer tidak dapat mempuyai konsep tak hingga sehingga pendekatan eksak di atas tidak akan digunakan. Sebaliknya, perhitungan akan digunakan dengan jumlah Riemann tertentu dengan partisi seimbang dengan norma tertentu, diimplementasi dengan *divide and conquer* dan *tag* dari tiap partisi adalah titik tengah dari interval.

Selain itu, sebelumnya sudah dijelaskan definisi dari fungsi yang terintegralkan Riemann. Sudah dapat dibuktikan bahwa fungsi polinom merupakan fungsi yang terintegralkan

Riemann. Maka penulis akan membatasi implementasi metode Riemann kepada fungsi polinom.

B. Fungsi Polinom

Dalam matematika, fungsi adalah suatu relasi yang memetakan tiap elemen pada domain secara unik ke salah satu elemen pada kodomain. Setiap elemen pada domain harus memiliki peta, yaitu suatu elemen di kodomain. Sebaliknya, elemen pada kodomain tidak harus memiliki pasangan di domain. Suatu fungsi dimana setiap elemen di kodomain memiliki pasangan di domain dinamakan fungsi surjektif. Selain itu, suatu fungsi dimana elemen berbeda di domain dipetakan ke elemen berbeda di kodomain dikatakan fungsi injektif. Suatu fungsi yang memiliki kedua sifat tersebut, surjektif dan injektif dinamakan fungsi bijektif.

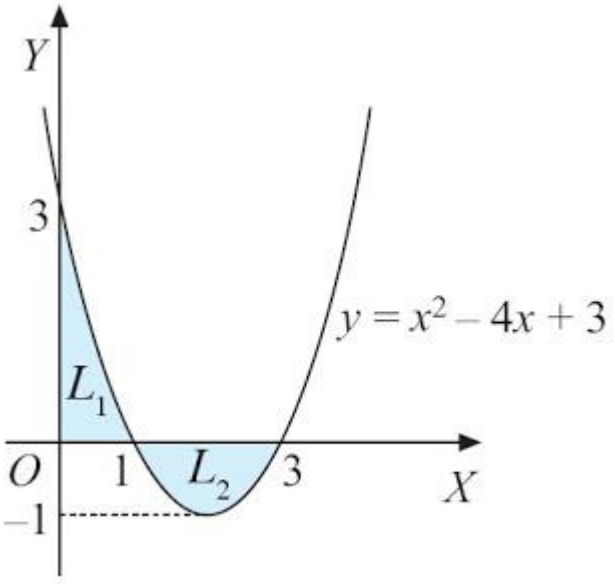
Contoh-contoh fungsi dinyatakan dalam bentuk eksplisit adalah $f(x) = x + 1$ atau $g(x) = \frac{2x+1}{x-1}$. Fungsi pertama dikatakan fungsi polinom. Sedangkan fungsi kedua dikatakan sebagai fungsi rasional. Fungsi yang akan dibahas selanjutnya adalah fungsi polinom.

Fungsi polinom atau polinomial atau suku banyak merupakan suatu fungsi yang dalam bentuk eksplisitnya melibatkan jumlah perkalian pangkat dalam satu atau lebih variabel dengan koefisien. Pangkat tertinggi dari suatu fungsi polinomial diberi nama derajat dari suatu polinom. Secara umum, setiap fungsi polinomial dapat dinyatakan dalam bentuk

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

dengan n merupakan derajat dari polinom dan $a_n \neq 0$.

Dengan definisi derajat di atas, dapat dilihat bahwa fungsi $f(x) = c$ merupakan polinom berderajat 0. Namun, terdapat kasus khusus, yaitu ketika $c = 0$, sehingga $f(x) = 0$, fungsi tersebut tidak dinamakan fungsi polinom berderajat 0, melainkan polinomial nol.



Gambar 3 Contoh kurva $f(x) = x^2 - 4x + 3$

Sumber : adityalojes.blogspot.co.id

Fungsi polinom merupakan contoh fungsi yang paling sederhana. Faktanya, hampir semua fungsi dapat dinyatakan dalam fungsi polinom dengan beberapa syarat dan kondisi tertentu. Misalkan fungsi eksponensial $f(x) = e^x$ dapat dihampiri dengan deret Taylor nya yang tidak lain merupakan fungsi polinom, namun dengan derajat tak hingga. Namun lagi-lagi, karena komputer tidak memiliki definisi tak hingga, maka bahasa-bahasa pemrograman yang mendukung fungsi-fungsi eksponensial seperti Matlab menggunakan aproksimasi dari deret Taylor e^x sampai derajat tertentu.

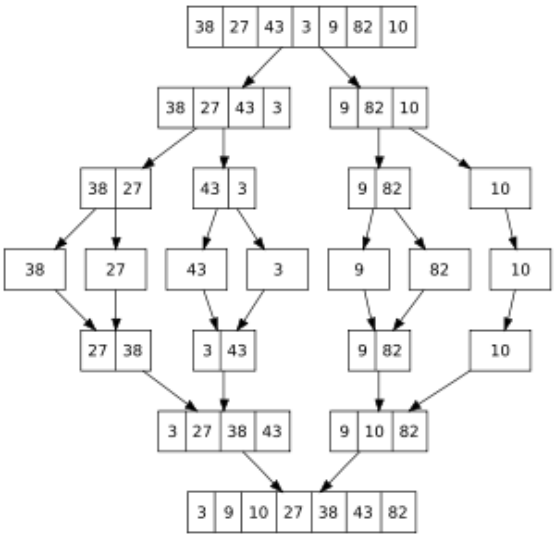
C. Divide and Conquer

Divide and Conquer merupakan suatu teknik dalam pemrograman dimana suatu persoalan dibagi-bagi menjadi sub-persoalan yang lebih kecil dan memecahkan tiap sub-persoalan secara terpisah sebelum akhirnya menggabung solusi dari tiap sub-persoalan agar dapat memecahkan persoalan secara utuh. Algoritma ini dapat menyelesaikan suatu persoalan lebih baik daripada *brute force* karena secara rata-rata memiliki kompleksitas yang jauh lebih baik dibandingkan *brute force*.

Secara umum, terdapat tiga bagian dari algoritma *divide and conquer*, yaitu :

1. *Divide*, adalah membagi persoalan menjadi beberapa sub-masalah yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil.
2. *Conquer*, atau disebut juga *solve*, yaitu memecahkan masing-masing sub-masalah secara rekursif.
3. *Combine*, yaitu menggabungkan solusi masing-masing sub-masalah sehingga membentuk solusi persoalan semula.

Mayoritas implementasi dari algoritma ini menggunakan metode rekursif, yaitu memanggil dirinya sendiri saat melakukan *divide* sampai dirasa sub-masalah yang didapat cukup trivial untuk dipecahkan,baru setelah itu sub-masalah yang sudah diselesaikan digabung untuk membentuk sub-masalah yang lebih besar namun seharusnya sudah lebih mudah.



Gambar 4 Ilustrasi sorting dengan *Divide and Conquer*

Sumber : chacancirebon.blogspot.co.id

Gambar di atas merupakan contoh ilustrasi pemecahan masalah *sorting* dengan algoritma *divide and conquer*. Persoalan di atas dapat diselesaikan dengan metode tersebut karena dapat dipecah menjadi sub-masalah yang memiliki tujuan sama dengan persoalan secara utuh, yaitu melakukan *sorting* terhadap tiap sub-masalah. Pada contoh di atas, kasus trivial adalah ketika panjang array 1, karena tidak perlu dilakukan apapun untuk melakukan *sorting* terhadap array dengan panjang 1.

Kompleksitas dari algoritma *divide and conquer* dapat ditulis dalam rumus sebagai berikut :

$$T(n) = \begin{cases} g(n) & , n \leq n_0 \\ 2T\left(\frac{n}{2}\right) + f(n) & , n > n_0 \end{cases}$$

Dengan

$T(n)$: waktu komputasi dengan ukuran masukan n.

$g(n)$: waktu komputasi untuk menyelesaikan sub-masalah *trivial.*

$f(n)$: waktu komputasi untuk menggabungkan solusi.

III. IMPLEMENTASI

Penulis akan mencoba melakukan implementasi dari algoritma *divide and conquer*, yaitu terhadap permasalahan menghitung luas daerah di bawah kurva sembarang fungsi, dengan pendekatan aproksimasi dengan metode jumlah Riemann. Implementasi kali ini akan dibatasi hanya untuk kasus khusus, yaitu suatu polinom dengan derajat berhingga. Hal ini karena polinom merupakan kasus dasar, karena hampir semua fungsi eksplisit juga dapat diaproksimasi dengan fungsi polinom berhingga. Sebelumnya, sudah dikatakan bahwa fungsi polinom terintegralkan Riemann, sehingga yang perlu dilakukan adalah mencari limit jumlah Riemann nya dengan aproksimasi sampai titik batas tertentu atau basis. Perlu dicatat pula bahwa jika kurva berada di bawah sumbu x maka hasil akan negatif.

Struktur data polinom dalam program dapat direpresentasikan dalam *array of coefficient* dan juga suatu integer untuk menampung derajatnya. Berikut contoh implementasi struktur data polinom dalam Java

```
1  public class Polinom {
2      private double [] data;
3      private int degree;
4
```

Gambar 5 Implementasi Polinom dalam Program

Sumber : Dokumen Pribadi

Perhitungan luas daerah dengan *divide and conquer* terhadap suatu fungsi polinom $f(x)$ pada selang $I = [a, b]$ dilakukan dengan cara berikut. Mula-mula, buat suatu partisi awal $P = \{a, b\}$. Dengan partisi tersebut, hanya ada satu interval, yaitu interval awal I. Lalu, tentukan suatu bilangan δ yang merupakan batas maksimum dari norma partisi. Semakin kecil nilai δ, akan semakin baik perhitungan program mendekati nilai sebenarnya dari luas daerah di bawah kurva. Jika norma dari partisi P masih lebih besar dari δ, tambahkan di tiap dua elemen berurutan, elemen baru yang merupakan titik tengah dari dua elemen tersebut untuk membentuk partisi baru P'. Contoh jika

$$P = \{a, b\} \rightarrow P' = \{a, \frac{a+b}{2}, b\}$$

dan jika

$$P = \{x_0, x_1, x_2\} \rightarrow P' = \{x_0, \frac{x_0 + x_1}{2}, x_1, \frac{x_1 + x_2}{2}, x_2\}$$

dan seterusnya.

Dengan konstruksi seperti di atas, maka setiap melakukan *divide*,

$$||P'|| = \frac{||P||}{2}$$

Dan pada akhirnya, akan mencapai kasus basis, yaitu ketika $||P|| < \delta$.

Setelah selesai melakukan pembagian permasalahan menjadi beberapa sub-masalah yang merupakan basis, hitung tiap blok partisi dengan memasukkan luas persegi panjang yaang dibentuk. Tag yang dipilih dari tiap interval adalah titik tengah dari tiap interval. Jadi, misalkan interval sudah dibagi menjadi interval kecil

$$I_n = [x_{n-1}, x_n]$$

Maka,

$$t_n = \frac{x_n - x_{n-1}}{2}$$

Tahap *conquer* dilakukan dengan menghitung luas persegi tiap interval, yaitu

$$f(t_n)(x_n - x_{n-1})$$

Tahap *combine* juga sangatlah mudah, yaitu dengan menambah setiap interval yang sudah dihitung untuk menyatukan semua interval. Setelah digabung semuanya, didapatlah hasil dari perhitungan luas daerah di bawah kurva.

Sebelumnya, untuk menghitung nilai dari $f(t_n)$, haruslah dibuat suatu fungsi yang memeriksa pemetaan dari suatu fungsi. Program untuk mendapat angka tersebut dapat dilihat dari contoh berikut

```
28      public double value(double val) {
29          double jwb = 0;
30          double temp = 1;
31          for (int i = 0; i <= degree; ++i) {
32              jwb += temp * data[i];
33              temp *= val;
34          }
35          return jwb;
36      }
```

Gambar 6 Implementasi Fungsi Menghitung Image dari Suatu Elemen Domain

Sumber : Dokumen Pribadi

Setelah memiliki fungsi untuk menghitung nilai dari suatu fungsi, akhirnya dapat dilakukan perhitungan untuk mencari luas daerah dibawah kurva suatu fungsi polinom dengan program sebagai berikut.

```
38      public double integral(double start, double end, double delta) {
39          if (end - start > delta) {
40              double mid = (start + end)/2;
41              double sumright = integral(mid, end, delta);
42              double sumleft = integral(start, mid, delta);
43              return sumright + sumleft;
44          } else {
45              return (end - start) * value((start + end)/2);
46          }
47      }
48  }
```

Gambar 7 Implementasi Menghitung Luas Daerah di Bawah Kurva

Sumber : Dokumen Pribadi

Dapat dilihat bahwa program di atas ekuivalen dengan metode *divide and conquer* yang sudah dijelaskan sebelumnya. Yaitu, partisi dilakukan di tengah dari setiap interval jika norma partisi masih lebih besar daripada nilai δ.

Berikut merupakan beberapa contoh hasil eksekusi program untuk fungsi $f(x) = x^2$. Diketahui bahwa

$$\int_0^1 f(x)\,dx = \frac{1}{3} = 0{,}33333\ldots$$

Sedangkan, menurut hasil program untuk nilai δ yang berbeda-beda adalah

```
C:\Users\W8.1\Desktop>java Main
Masukkan titik awal : 0
Masukkan titik akhir : 1
Masukkan delta maksimum : 0,01
0.3333282470703125

C:\Users\W8.1\Desktop>java Main
Masukkan titik awal : 0
Masukkan titik akhir : 1
Masukkan delta maksimum : 0,0001
0.3333333330228925

C:\Users\W8.1\Desktop>java Main
Masukkan titik awal : 0
Masukkan titik akhir : 1
Masukkan delta maksimum : 0,000001
0.33333333333325754

C:\Users\W8.1\Desktop>
```

Gambar 8 Hasil Eksekusi Program

Sumber : Dokumen Pribadi

Dapat dilihat bahwa semakin kecil nilai δ, maka semakin akurat pula hasil aproksimasi yang dihasilkan oleh program. Semakin kecil, nilai dari luas di bawah kurva dengan metode Riemann untuk $f(x) = x^2$ untuk interval $[0,1]$ semakin mendekati nilai aslinya, yaitu 0,33333 .....

Berikut akan dilakukan juga percobaan untuk fungsi dengan derajat yang lebih tinggi, yaitu $g(x) = x^3 - x^2 + 4x + 10$, dan akan dilakukan percobaan untuk interval $[1,6]$ untuk beberapa nilai δ, dan akan dilihat perbandingannya. Dimana diketahui

$$\int_1^6 x^3 - x^2 + 4x + 10\,dx = \frac{4465}{12} = 372{,}08333\ldots$$

```
C:\Users\W8.1\Desktop>java Main
Masukkan titik awal : 1
Masukkan titik akhir : 6
Masukkan delta maksimum : 0,005
372.08323895931244

C:\Users\W8.1\Desktop>java Main
Masukkan titik awal : 1
Masukkan titik akhir : 6
Masukkan delta maksimum : 0,00005
372.0833333275732

C:\Users\W8.1\Desktop>java Main
Masukkan titik awal : 1
Masukkan titik akhir : 6
Masukkan delta maksimum : 0,0000005
372.083333333333

C:\Users\W8.1\Desktop>
```

Gambar 9 Hasil Eksekusi Program

Sumber : Dokumen Pribadi

Sama seperti hasil sebelumnya, nilai dari hasil aproksimasi semakin mendekati hasil aslinya, yaitu 372,083333... ketika nilai dari δ semakin mendekati 0. Bahkan dengan nilai $\delta = 5 \times 10\text{^} - 7$ angka di belakang koma dari nilai sudah menyerupai nilai aslinya untuk representasi bilangan desimal di Java.

IV. KESIMPULAN

Dari hasil eksekusi program implementasi yang telah dibuat oleh penulis, dapat disimpulkan bahwa komputer tidak dapat benar-benar menerapkan konsep ketakterhinggaan, sehingga perlu adanya suatu metode penghampiran untuk melakukan komputasi terhadap permasalahan-permasalahan yang pada bidang matematika menggunakan konsep ketakterhinggan, terutama kalkulus, seperti turunan dan integral.

Konsep integral sangat erat kaitannya dengan perhitungan luas daerah di bawah kurva, dan melibatkan limit jumlah dengan partisi dimana norma nya merupakan suatu bilangan *infinitesimal.* Namun, karena keterbatasan komputer, metode jumlah Riemann digunakan untuk menghitung luas daerah di bawah kurva, dan dari pengamatan, terbukti bahwa semakin kecil norma partisi, maka akan semakin kecil pula galat dari perhitungan yang dihasilkan.

V. SARAN

Penulis menggunakan bahasa Java untuk melakukan implementasi program aproksimasi luas daerah di bawah kurva di atas. Sebenarnya, bahasa Java bukanlah salah satu bahasa yang di-*design* untuk melakukan komputasi maupun analisis numerik seperti yang telah dilakukan penulis. Namun, karena keterbatasan waktu, maka penulis hanya melakukan implementasi pada bahasa Java. Selain itu, penulis hanya melakukan percobaan terhadap fungsi paling sederhana, yaitu fungsi polinom. Untuk kedepannya, mungkin dapat dibuat juga program untuk melakukan pendekatan dengan *divide and*

conquer pada fungsi-fungsi lain, seperti fungsi eksponensial, logaritma, trigonometri, bahkan fungsi-fungsi pada bilangan kompleks.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa, berkat rahmat-Nya penulis dapat menyelesaikan makalah ini. Juga kepada dosen Strategi Algoritma IF ITB yang telah membimbing selama satu semester sehingga saya mendapat banyak pengetahuan dari mata kuliah ini, yaitu kepada Ir. Rinaldi Munir. Juga semua orang yang telah membantu proses penyelesaian makalah ini sehingga penulis dapat menyelesaikan makalah ini dengan tepat waktu.

REFERENCES

[1] Bartle, Robert G. 2000. *Introduction to Real Analysis*. 3rd ed.

[2] Munir, Rinaldi. 2006. *Strategi Algoritma*. Bandung : Penerbit Informatika

[3] www.math.wisc.edu diakses 18 Mei 2017 pada 17.15

[4] https://revisionmaths.com diakses 18 Mei 2017 pada 13.10

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.

Bandung, 29 April 2012

Dewita Sonya Tarabunga

13515021