Proses Pencarian Akar Persamaan Matematika Menggunakan Metode Bagi-Dua

Devin Hoesen - 13510081
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510081@std.stei.itb.ac.id

***Abstrak* – Makalah ini membahas penggunaan desain algoritma *Decrease and Conquer* untuk mencari solusi suatu persamaan matematika satu variabel dalam bentuk Metode Bagi-Dua. Metode ini menggunakan *Decrease and Conquer*, yaitu memecah-mecah permasalahan menjadi submasalah-submasalah kemudian memproses salah satu submasalah. Metode ini menjamin terperolehnya solusi persamaan dengan syarat-syarat tertentu, di antaranya selang awal yang tepat. Itulah salah satu kelebihan metode ini. Namun, terdapat juga beberapa kelemahan metode ini. Metode ini beranalogi dengan Pencarian Biner pada larik, hanya saja di sini, data yang diproses adalah data yang kontinu. Beberapa catatan mengenai implementasi metode ini dalam penghitungan di komputer juga dibahas. Terakhir, diberikan contoh mengenai penggunaan metode pencarian akar ini untuk mencari akar persamaan matematika menggunakan aplikasi pengolah angka (seperti *Microsoft Office Excel*).**

***Istilah Kunci* – Akar, *Decrease and Conquer*, Metode Bagi-Dua, Persamaan Matematika, Teorema Nilai Antara**

***Abstract* – This paper is about the use of Decrease and Conquer algorithm design for finding one-variable mathematical-equation's root in form of Bisection Method. This method uses Decrease and Conquer, that is, reducing problem to subprolems, then processing one instance of subproblem. This method is guaranteed to converge to equation's root with some preconditions, one of them is the right initial interval. That is one of advantages of this method. However, there are some disadvantages. This method is analogous with Binary Search in array. The difference is Bisection Method processes continuous data. Some notes of implementation of this method in computer is also discussed here. Lastly, an example of application of this root-finding method to find mathematical-equation's root by using spreadsheet application (like *Microsoft Office Excel*).**

***Index Terms* – Bisection Method, Decrease and Conquer, Intermediate Value Theorem, Mathematical Equation, Root**

I. DASAR TEORI

A. *Algoritma* Divide and Conquer

Metode ini dikenal pada zaman penjajahan dengan nama *devide et impera* atau pecah-belahlah dan taklukkan. Pada saat itu, negara-negara penjajah memecah-belah negara yang dijajahnya hingga kekuatannya lemah. Para penjajah lalu menaklukkan negara jajahannya tersebut dengan mudah.

Serupa dengan hal tersebut, pada desain algoritma, algoritma *divide and conquer* memecah-mecah masalah menjadi submasalah-submasalah yang penyelesaiannya serupa dengan masalah asal namun lebih mudah diselesaikan daripada masalah asal karena ukuran masalahnya lebih kecil.

Algoritma ini telah digunakan sejak lama dan sangat sering digunakan terutama dalam pengurutan larik. Salah satu contohnya adalah *merge-sort*. Pada pengurutan jenis ini, larik dipecah hingga menjadi sublarik-sublarik beranggota satu elemen. Larik beranggota satu elemen ini dianggap sudah terurut. Sublarik-sublarik ini kemudian digabungkan secara berulang-ulang menjadi larik terurut hingga terbentuk satu buah larik, yakni larik semula yang sudah terurut.

Algoritma ini bekerja sebagai berikut.

(1) *Divide*, membagi masalah menjadi beberapa submasalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
(2) *Conquer*, memecahkan (menyelesaikan) masing-masing submasalah, dan
(3) *Combine*, mengabungkan solusi masing-masing submasalah sehingga membentuk solusi masalah semula.

Dari cara kerja algoritma ini yang memecah-mecah persoalan menjadi lebih kecil cakupannya, secara natural, algoritma ini lebih tepat diimplementasikan secara rekursif.

Ada beberapa keuntungan dari algoritma ini, di antaranya adalah sebagai berikut.

(1) Algoritma ini dapat memecahkan beberapa masalah yang sulit. *Divide and conquer* adalah alat yang sangat efektif untuk memecahkan beberapa masalah yang secara konsep sangat sulit.
(2) Algoritma ini dapat menjadi alat bantu untuk menemukan algoritma yang efisien.
(3) Algoritma ini cenderung menggunakan *cache* memori secara efisien. Ketika submasalah sudah cukup kecil, seluruh submasalah, secara prinsip, dapat diselesaikan di dalam *cache*, tanpa perlu mengakses memori utama. Lebih jauh, *divide and conquer* dapat didesain agar algoritma-algoritma penting (seperti pengurutan larik dan pengalian matriks) dapat memanfaatkan *cache* secara lebih optimal.

(4) Dalam komputasi dengan dengan pembulatan, yakni dengan penggunaan bilangan titik-mengambang, '*floating-point*', algoritma ini dapat memberikan hasil yang lebih akurat daripada metode iteratifnya yang setara.

B. Algoritma Decrease and Conquer

Algoritma *Decrease and Conquer* serupa seperti *Divide and Conquer*. Perbedaannya adalah jika pada algoritma *Divide and Conquer* semua submasalah diproses, pada *Decrease and Conquer* hanya satu submasalah saja yang diproses di setiap tahap untuk memperleh solusi masalah.

Algoritma ini pun telah lama digunakan dan umumnya digunakan dalam algoritma pencarian, meskipun ada pula algoritma pengurutan larik yang menggunakan *Decrease and Conquer*. Salah satu contoh algoritma yang menggunakan *Decrese and Conquer* sebagai subalgoritmanya adalah Algoritma Euclid dan pencarian biner. Algoritma Euclid digunakan untuk menemukan faktor persekutuan terbesar (FPB) di antara dua bilangan. Algoritma Euclid mereduksi bilangan-bilangan yang ingin dicari FPB-nya hingga ditemukan FPB-nya Sementara itu, pencarian biner dilakukan pada larik terurut membesar ataupun mengecil untuk mencari suatu bilangan pada larik tersebut. Larik dibagi dua dan bilangan yang dicari ditentukan masuk bagian yang mana. Proses tersebut dilakukan secara berulang hingga bilangan yang dicari ditemukan.

Proses penentuan akar suatu persamaan matematika dengan Metode Bagi-Dua beranalogi dengan pencarian biner tersebut. Perbedaannya, bila pada pencarian biner, selang pencarian berupa data diskret, pada Metode Bagi-Dua, selang pencarian merupakan data kontinu.

Algoritma ini bekerja sebagai berikut.

(1) *Decrease*, memperkecil cakupan masalah menjadi beberapa masalah yang lebih kecil (submasalah),
(2) *Conquer*, memproses salah satu subpersoalan di setiap tahap.

Tidak ada proses *Combine* pada algoritma ini seperti pada algoritma *Divide and Conquer* karena hanya satu submasalah saja yang diproses.

Ada tiga tipe algoritma *Decrease and Conquer*, yaitu sebagai berikut.

(1) *Decrease by a constant*, ukuran instans submasalah direduksi berdasarkan konstanta yang sama pada setiap iterasi/rekursi algoritma, biasanya dengan konstanta 1. Contoh tipe ini misalnya:
   - pengurutan dengan insersi (*Insertion Sort*),
   - pengurutan dengan seleksi (*Selection Sort*),
   - *Breadth-First Search*,
   - *Depth-First Search*.
(2) *Decrease by a constant factor*, ukuran instans submasalah direduksi berdasarkan faktor konstanta yang sama pada setiap iterasi/rekursi algoritma, biasanya dengan faktor konstanta 2. Contoh tipe ini misalnya:
   - Pencarian Biner (*Binary Search*),
   - Metode Bagi-Dua (*Bisection Method*),
   - persoalan pencarian koin palsu.
(3) *Decrease by a variable size*, pola reduksi instans submasalah bervariasi pada setiap iterasi/rekursi algoritma. Contoh tipe ini misalnya Algoritma Euclid.

C. Suku Banyak dan Metode Horner

Suku banyak (polinom) adalah ekspresi dengan panjang terbatas yang dibangun dari variabel dan konstanta, dengan hanya menggunakan operasi penambahan, pengurangan, pengalian, dan pangkat bilangan-bulat tak-negatif. Pembagian dengan konstanta diperbolehkan dalam suku banyak. Pangkat tertinggi variabel dalam suku banyak disebut dengan derajat suku banyak tersebut. Kata "polinom" berasal dari bahasa Yunani *poli* 'banyak' dan bahasa Latin *binomium* 'binomial'. Kata "polinom" diperkenalkan ke dalam bahasa Latin oleh Fransiscus Vieta.

Metode Horner merupakan algoritma untuk menghitung suku banyak satu-variabel dengan mengubah suku banyak tersebut menjadi bentuk yang lebih efisien untuk dikomputasi. Metode ini dinamai dari matematikawan Inggris William George Horner.

Diberikan suatu suku banyak

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n$$

dengan a_0, …, a_n merupakan bilangan nyata. Lalu, kita definisikan serangkaian konstanta baru sebagai berikut.

$$\begin{aligned} b_n &:= a_n \\ b_{n-1} &:= a_{n-1} + b_n x_0 \\ &\vdots \\ b_0 &:= a_0 + b_1 x_0 \end{aligned}$$

$p(x_0)$ kemudian dapat dituliskan dalam bentuk sebagai berikut.

$$\begin{aligned} p(x_0) &= a_0 + x_0(a_1 + x_0(a_2 + ... + x_0(a_{n-1} + a_n x_0)...)) \\ &= a_0 + x_0(a_1 + x_0(a_2 + ... + x_0(a_{n-1} + b_n x_0)...)) \\ &= a_0 + x_0(a_1 + x_0(a_2 + ... + x_0(b_{n-1})...)) \\ &\vdots \\ &= a_0 + x_0(b_1) \\ &= b_0 \end{aligned}$$

Dari persamaan terakhir dapat disimpulkan bahwa b_0 merupakan nilai dari $p(x_0)$.

Dengan menggunakan pembagian sintetis, metode Horner ini dapat divisualisasikan sebagai berikut.

	x^n	x^{n-1}	...	x^0
x_0 |	a_n	a_{n-1}	...	a_0
|		$a_n x_0$	...	$a_1 x + ... + a_{n-1} x^{n-1} + a_n {x_0}^n$
	a_n	$a_{n-1} + a_n x_0$	...	$a_0 + ... + a_{n-1} x^{n-1} + a_n {x_0}^n$

Suku-suku pada barisan terakhir merupakan jumlah dari suku-suku pada dua baris di atasnya. Suku-suku pada baris kedua terakhir merupakan hasil kali x_0 dengan suku yang tepat berada di sebelah kirinya pada baris terakhir

(suku di kiri-bawahnya). Terlihat bahwa suku terakhir pada baris terakhir merupakan nilai dari $p(x_0)$.

Metode Horner ini bisa mengubah penghitungan suku banyak menjadi lebih efisien bila dilakukan pada komputer. Bila penghitungan suku banyak dengan pemangkatan biasa membutuhkan berkali-kali pengalian pada setiap tahapnya, metode Horner hanya membutuhkan sekali pengalian saja setiap tahapnya. Pertimbangkan kode-semu berikut ini.

```
function Horner(degree:integer, a:array of
   integer, x0:real) -> real
{a[0] merupakan a_n dst.}

DEKLARASI VARIABEL LOKAL
result : real
i : integer

ALGORITMA
result <- a[0];
i <- 1;
repeat degree times
   result <- result * x0 + a[i];
   i <- i + 1;
return result;
```

Dari kode-semu tersebut, dengan n sebagai derajat suku banyak, terdapat dua kali operasi pengisian nilai, n kali operasi pengalian, penambahan dan pengisian nilai, dan n kali operasi penambahan dan pengisian nilai. Dengan demikian $T(n) = 5n + 2 = O(n)$. Bila dilakukan dengan pemangkatan biasa, akan terdapat $\frac{n}{2}(n+1)$ perkalian dan n kali penambahan, membuat $T(n) = \frac{n^2}{2} + \frac{3n}{2} = O(n^2)$. Jelas metode Horner lebih efisien daripada pemangkatan biasa.

II. METODE BAGI-DUA

A. Teorema Nilai Antara

Metode Bagi-Dua bermula dari Teorema Nilai Antara. Teorema ini hanya berlaku bagi fungsi yang kontinu. Fungsi f yang kontinu pada selang $[a,b]$ berarti fungsi tersebut tidak memiliki lompatan; fungsi tersebut bisa kita "gambar" dari titik $(a, f(a))$ ke $(b, f(b))$ tanpa harus mengangkat pensil kita dari kertas. Akibatnya, fungsi f harus bisa memiliki semua nilai di antara $f(a)$ dan $f(b)$. Sifat ini dinyatakan dalam teorema berikut.

Teorema Nilai Antara - Misalkan f merupakan fungsi yang terdefinisi pada selang $[a,b]$ dan misalkan ada bilangan W yang berada di antara $f(a)$ dan $f(b)$. Jika f merupakan fungsi yang kontinu pada selang $[a,b]$, maka ada sedikitnya satu bilangan c di antara a dan b yang memenuhi $f(c) = W$.

Hal yang paling penting dari teorema ini adalah kekontinuan fungsi. Artinya, teorema ini tidak bisa dipakai ketika fungsi tidak kontinu pada selang $[a,b]$ yang dimaksud. Untungnya fungsi yang ingin dicari akar persamaannya kebanyakan kontinu.

Kebalikan dari teorema ini tidak sepenuhnya benar. Katakanlah f bisa memiliki semua nilai di antara $f(a)$ dan $f(b)$, maka kita tidak bisa menjamin bahwa f kontinu. Hanya karena sebuah fungsi memiliki nilai antara, tidak berarti fungsi tersebut bisa dijamin kekontinuannya.

Teorema Nilai Antara ini bisa dipakai untuk memberi tahu kita sesuatu mengenai solusi sebuah persamaan, seperti yang bagian berikutnya jelaskan.

B. Prinsip Kerja Metode Bagi-Dua

Teorema Nilai Antara bisa dipakai untuk menghampiri solusi dari persamaan $f(x) = 0$ dengan secara berulang-ulang membagi dua selang yang diketahui memiliki solusi persamaan. Metode Bagi-Dua ini memiliki dua keuntungan, yakni kemudahannya untuk dimengerti dan keandalannya.

Algoritma Metode Bagi-Dua

Misalkan $f(x)$ merupakan fungsi kontinu dan misalkan a_1 dan b_1 memenuhi $a_1 < b_1$ dan $f(a) \cdot f(b) < 0$. Misalkan E menyatakan galat yang diinginkan $|r - m_n|$. Ulangi langkah 1 s.d. 5 berikut untuk $n = 1,2,3,\ldots$ hingga $h_n < E$.

1. Hitung $m_n = \frac{a_n + b_n}{2}$.
2. Hitung $f(m_n)$ dan bila $f(m_n) = 0$, BERHENTI.
3. Hitung $h_n = \frac{b_n - a_n}{2}$.
4. Jika $f(a_n) \cdot f(m_n) < 0$, pilih $a_{n+1} = a_n$ dan $b_{n+1} = m_n$.
5. Jika $f(a_n) \cdot f(m_n) > 0$, pilih $a_{n+1} = m_n$ dan $b_{n+1} = b_n$.

Mulai prosesnya dengan mensketsa grafik dari f yang merupakan fungsi kontinu. Akar real r dari $f(x) = 0$ adalah koordinat x ketika $f(x)$ memotong sumbu-x. Langkah pertama untuk mencari solusi persamaan adalah dengan memilih dua titik $a_1 < b_1$ yang kita yakin membuat f memiliki nilai yang berlawanan tanda; jika f memiliki nilai yang berbeda tanda pada a dan b maka hasil kali dari $f(a) \cdot f(b)$ merupakan bilangan negatif. Teorema Nilai Antara menjamin adanya akar persamaan di antara a_1 dan b_1. Sekarang hitunglah nilai f di titik tengah $m_1 = \frac{a_1 + b_1}{2}$ dari $[a_1,b_1]$. Bilangan m_1 merupakan hampiran pertama kita ke r.

Sekarang akan terjadi beberapa kasus. Kasus pertama, $f(m_1) = 0$, dengan begitu proses selesai dan kita mendapati akar $r = m_1$. Jika tidak, $f(m_1)$ pasti berbeda tanda dengan salah satu dari $f(a)$ atau $f(b)$. Tentukan subinterval baru $[a_2,b_2]$ dari salah satu subinterval $[a_1,m_1]$ atau $[m_1,b_1]$ yang mengandung perubahan tanda dari f. Hitung kembali titik tengah dari selang ini, $m_2 = \frac{a_2 + b_2}{2}$. Bilangan m_2 merupakan hampiran kedua kita ke r.

Ulangi proses ini, yakni menentukan sederetan hampiran m_1, m_2, m_3, … dan subinterval $[a_1,b_1]$, $[a_2,b_2]$, $[a_3,b_3]$, …, setiap subinterval mengandung akar r dan panjang selangnya setengah dari panjang subinterval sebelumnya. Berhentilah ketika r sudah berada dalam

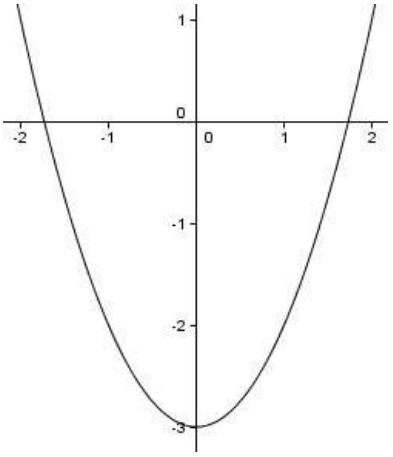
akurasi yang diinginkan, yaitu ketika $\frac{b_n - a_n}{2}$ lebih kecil daripada galat yang diizinkan (dinotasikan dengan *E*).

C. Analisis Terhadap Metode Bagi-Dua

Ada beberapa kelebihan metode ini, salah satunya metode ini pasti konvergen pada selang [*a*,*b*] jika *f* kontinu pada selang tersebut serta *f*(*a*) dan *f*(*b*) berbeda tanda. Galat mutlaknya berkurang setengahnya setiap tahap.

Namun ada juga beberapa kelemahan metode ini. Pertama, metode ini konvergen secara linear. Meskipun metode ini menjamin terperolehnya solusi jika selangnya tepat, metode ini memusat ke solusi dengan lambat.

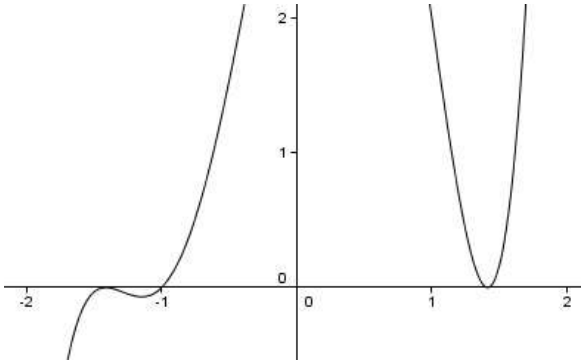
Kedua, pemilihan selang haruslah tepat. Jika selang terlalu lebar ataupun terlalu sempit, solusi mungkin saja tidak diperoleh meskipun persamaan tersebut memiliki solusi. Perhatikanlah grafik fungsi berikut ini.



Gambar 1 – Grafik f(x) = $x^2 - 3$

Bila kita memilih selang [-2,2] ataupun [-1,1], Metode Bagi-Dua langsung gagal memusat ke solusi persamaan pada hampiran pertama, karena $f(a) \cdot f(b) > 0$. Namun bila kita memilih selang [-2,0] ataupun [0,2], metode ini dijamin memusat ke solusi persamaan setelah sekian kali iterasi.

Ketiga, ada kalanya muncul akar kembar pada solusi persamaan suku banyak. Bila ini terjadi, Metode Bagi-Dua pasti gagal menemukan akar kembar ini. Ini merupakan kelemahan terbesar Metode Bagi-Dua ini. Perhatikanlah grafik fungsi berikut.



Gambar 2 – Grafik fungsi f(x) = $(x + 1)(x^2 - 2)^2$

Akar persamaan $f(x) = 0$ yang berada pada selang [1,2] dan [-2,-1) pasti akan gagal dideteksi oleh metode ini karena di sebelah kiri dan kanan akar pada selang-selang tersbut, nilai-nilai *x* tidak berubah tanda; grafik tidak memotong sumbu-*x* namun hanya "menyentuhnya" saja.

Keempat, metode ini hanya dapat menemukan satu solusi pada satu selang tertentu. Dalam satu selang, pasti terdapat sejumlah ganjil solusi agar metode ini dapat konvergen (tanpa memperhatikan akar kembar). Bila terdapat sejumlah genap solusi dalam suatu selang, metode ini pasti tidak konvergen karena batas-batas selang pasti tidak membuat nilai fungsi berbeda tanda. Akibatnya, jila terdapat lebih dari satu solusi dalam satu selang, sejumlah dua atau kelipatan dua solusi pasti tidak dapat ditemukan. Agar seluruh solusi dapat ditemukan, bisa dilakukan sejumlah upaya seperti yang akan dijelaskan pada bagian berikut.

D. Beberapa Catatan Mengenai Implementasi Metode Bagi-Dua Pada Komputer

Metode ini melibatkan penghitungan nilai fungsi secara berulang-ulang. Untuk penghitungan suku banyak satu-variabel, proses tersebut dapat diefisienkan dengan menggunakan metode Horner. Seperti telah dijelaskan sebelumnya, metode Horner lebih efisien daripada cara pemangkatan biasa dengan perbandingan $O(n)$ berbanding $O(n^2)$. Selain itu, kebanyakan persamaan matematika berbentuk suku banyak satu-variabel.

Akar suatu persamaan dapat dicari satu-satu dengan *brute-force* atau dengan *divide and conquer* untuk menemukan selang-selang yang mengandung akar-akar persamaan. *Brute-force* bekerja dengan mengunjungi selang satuan dimulai dari 0 hingga suatu batas tertentu ke arah sumbu positif dan negatif. Jika batas-batas suatu selang satuan tersebut ternyata membuat fungsi *f* berbeda tanda, selang satuan tersebut bisa dipakai sebagai selang awal untuk memulai proses Metode Bagi-Dua.

Pencarian seluruh akar satu-satu juga dapat dilakukan dengan metode *divide and conquer*. Bila ingin memperoleh selang satuan yang mengandung akar persamaan, kita bisa mulai dari selang yang panjangnya merupakan hasil pemangkatan dari dua ke kiri dan ke kanan dari 0 yang lebih besar atau sama dengan nilai tertentu yang diperkirakan mengandung seluruh akar persamaan. Untuk mengefisienkan pencarian eksponen dari 2 yang tepat ini, bisa kita gunakan metde Horner. Kemudian, selang tersebut terus dibagi dua, selang yang batas-batasnya menyebabkan *f* berbeda tanda dimasukkan ke dalam himpunan selang solusi untuk kemudian dipecah kembali. Pemecahan selang ini berhenti hingga mencapai selang satuan. Proses ini mirip dengan melakukan algoritma *Decrease and Conquer* secara simultan, namun karena dapat memproses lebih dari satu submasalah, algoritma yan dipakai adalah *Divide and Conquer* bukan *Decrease and Conquer*. Jika ditemukan beberapa selang satuan yang diperkirakan mengandung solusi persamaan, pada setiap selang itu kemudian dilakukan *Decrease and Conquer* satu-satu.

Tabel I – Proses Penghampiran Menggunakan Metode Bagi-Dua

n	a_n	b_n	m_n	$f(a_n) * f(m_n)$	$f(m_n) * f(b_n)$
1	1	5	3	0.11874839215823500000	-0.13532340136926400000
2	3	5	4	-0.10679997423758200000	0.72571628387640800000
3	3	4	3.5	-0.04950253191882540000	0.26547362202767300000
4	3	3.5	3.25	-0.01526849825692730000	0.03795303851078410000
5	3	3.25	3.125	0.00234144796513402000	-0.00179516201186336000
6	3.125	3.25	3.1875	-0.00076142223081490300	0.00496520707574964000
7	3.125	3.1875	3.15625	-0.00024318440425422800	0.00067261947207858100
8	3.125	3.15625	3.140625	0.00001605520157163350	-0.00001418272381365830
9	3.140625	3.15625	3.1484375	-0.00000662338744675132	0.00010032290925758100
10	3.140625	3.1484375	3.14453125	-0.00000284353882903408	0.00002011405507599820
11	3.140625	3.14453125	3.14257813	-0.00000095359464456032	0.00000289589811181452
12	3.140625	3.14257813	3.14160156	-0.00000000862073759726	0.00000000877947488371
13	3.140625	3.14160156	3.14111328	0.00000046386627529212	-0.00000000427068496720
14	3.14111328	3.14160156	3.14135742	0.00000011276357215526	-0.00000000209565820552
15	3.14135742	3.14160156	3.14147949	0.00000002661915040754	-0.00000000100814476974
16	3.14147949	3.14160156	3.14154053	0.00000000589867908327	-0.00000000046438804520
17	3.14154053	3.14160156	3.14157104	0.00000000112637873997	-0.00000000019250968215
18	3.14157104	3.14160156	3.1415863	0.00000000013721242350	-0.00000000005657050055
19	3.1415863	3.14160156	3.14159393	-0.00000000000812476953	0.00000000001139909026
20	3.1415863	3.14159393	3.14159012	0.00000000001609809597	-0.00000000000324380458

Bagi suku banyak satu-variabel, batas tertentu untuk menghentikan algoritma *brute-force* tersebut adalah $\pm a_0$ (konstanta/suku yang tidak mengandung variabel). Untuk algoritma *Divide and Conquer*, nilai tertentu yang diperkirakan mengandung seluruh akar persamaan juga adalah $\pm a_0$. Bila persamaan yang ingin diselesaikan adalah persamaan trigonometri, kelipatan $\pm 2\pi$ bisa dijadikan batas tersebut.

Satu hal yang perlu diperhatikan dalam kedua algoritma tersebut adalah jarak antarakar persamaan minimal 1 satuan. Bila ada dua akar yang jaraknya kurang dari 1 satuan, kedua akar tersebut bisa saja tidak terdeteksi. Bisa saja selang yang dicari diperkecil misalnya menjadi setengahan atau seperempatan, namun dengan akibat pencarian selang yang tepat menjadi lebih lama dan tidak efisien.

Metode Bagi-Dua yang dibantu dengan algoritma pencarian akar yang telah dijelaskan sebelumnya bisa menemukan akar kembar asalkan akar kembar tersebut merupakan bilangan bulat.

E. Contoh Penggunaan Metode Bagi-Dua

Tabel I merupakan contoh penggunaan Metode Bagi-Dua untuk menghampiri solusi dari persamaan $f(x) = \sin x = 0$ pada selang [1,5] dengan 20 iterasi. Tabel I dihasilkan dengan menggunakan bantuan perangkat lunak *Microsoft Office Excel*.

Terlihat bahwa hampiran terakhir memberikan $m_n = 3.141590118$. Solusi sebenarnya persamaan tersebut adalah $x = \pi = 3.1415926535\ldots$ Hampiran setelah 20 kali iterasi mempunyai akurasi hingga lima angka desimal. Seperti telah dijelaskan, metode ini konvergen dengan lambat seperti ditunjukkan oleh Tabel I.

III. KESIMPULAN

Metode Bagi-Dua merupakan metode pencarian akar yang sederhana dan andal karena pasti konvergen bila selang yang dipilih tepat. Metode ini menggunakan desain algoritma *Decrease and Conquer*. Ada beberapa kelebihan metode ini. Pertama, sederhana. Kedua, pasti konvergen ke solusi bila selangnya tepat. Ketiga, hampiran yang didapat akurat, seperti kelebihan pada desain algoritma *Divide and Conquer* dan desain turunannya, yakni *Decrease and Conquer*. Namun, metode ini memiliki beberapa kelemahan. Pertama, konvergen secara linear dan bila dibandingkan dengan metode pencarian akar lain cenderung lebih lambat. Kedua, sulit untuk menemukan akar kembar kecuali bila akar kembar tersebut bilangan bulat. Ketiga, hanya bisa menemukan satu akar untuk satu selang. Keempat, pemilihan selang haruslah tepat.

Implementasinya di komputer disarankan memperhatikan beberapa catatan. Pertama, untuk suku

banyak satu-variabel, penghitungan nilai suku banyak bisa dilakukan dengan metode Horner. Hal ini agar penghitungan nilai suku banyak lebih efisien karena Metode Bagi-Dua melibatkan banyak sekali penghitungan nilai suku banyak. Kedua, untuk menemukan lebih dari satu solusi, pencarian selang awal yang mengandung solusi bisa dilakukan dengan *brute-force* atau dengan *Divide and Conquer*.

REFERENSI

[1] Dale Varberg, Edwin J. Purcell, dan Steven E. Rigdon. *Calculus – Ninth Edition*. New Jersey: Pearson Education, Inc. 2007.

[2] Rinaldi Munir. *Diktat Kuliah IF3051 – Strategi Algoritma*. Bandung: Program Studi Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. 2009.

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, 18 Desember 2012

Devin Hoesen
13510081