LAPORAN MAKALAH IF-5162 Metode Numerik Lanjut

Studi Kasus Mencari Invers Suatu Bilangan

Dipersiapkan oleh:

Gugy Lucky Khamdani / 23517041

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Sekolah Teknik Elektro dan Informatika ITB	**Nomor Dokumen**		**Halaman**
	IF5162/23517041		*17*
	-	-	*23 Mei 2018*

Daftar Isi

Daftar Gambar

Daftar Tabel

1 Persoalan

Diberikan sebuah permasalahan mengenai pencarian invers dari suatu bilangan. Banyak *supercomputer* tidak memiliki unit untuk membagi suatu bilangan dalam pengimplementasiannya. Hal ini disebabkan oleh operasi pembagian biasanya memerlukan sekitar 20-25 *clock cycle* untuk mendapatkan hasil pembagiannya, dan operasi tersebut lima kali lebih banyak dari *clock cycle* yang dibutuhkan untuk melakukan operasi perkalian. Untuk mengatasi masalah ini, sebagai gantinya dikembangkan cara untuk melakukan pembagian dengan cara menyelesaikan persamaan nirlanjar secara numerik, yang memungkinkan proses pembagian menjadi lebih cepat.

Untuk melakukan pembagian, persamaanya bisa dirumuskan sebagai

$$\frac{b}{a} = b \times \frac{1}{a} = b \times c$$

yang dalam hal ini,

$$c = \frac{1}{a}$$

sehingga untuk mendapatkan hasil dari pembagian b/a yang perlu dilakukan hanya melakukan perkalian $b \times c$. Permasalahannya adalah bagaimana cara mendapatkan nilai c tanpa harus menggunakan unit pembagian.

2 Landasan Teori

2.1 Solusi Persamaan Nirlanjar

Persoalan mencari solusi persamaan nirlanjar dapat dirumuskan secara singkat sebagai berikut: tentukan nilai x yang memenuhi persamaan

$$f(x) = 0$$

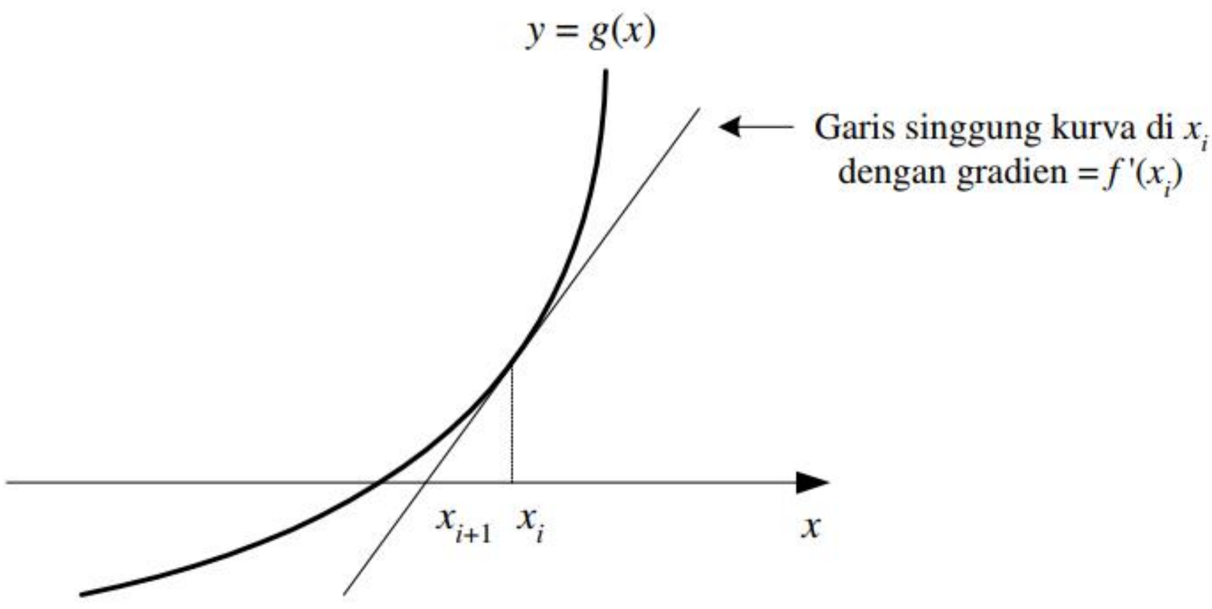
yaitu nilai $x = s$ sedemikian sehingga $f(s)$ sama dengan 0. Solusi dari persamaan ini lazim disebut sebagai akar-akar persamaan (*roots of equation*) atau nilai-nilai nol. Umumnya persamaan yang akan dipecahkan mucul dalam bentuk nirlanjar (*nonlinear*) yang melibatkan bentuk seperti sinus, cosinus, eksponensial, logaritma, dan fungsi transenden lainnya yang rumit atau kompleks sehingga tidak dapat dipecahkan secara analitik. Bila metode analitik tidak dapat menyelesaikan persamaan, maka solusinya masih bisa dicari denga enggunakan metode numerik.

Dalam metode numerik, pencarian akar $f(x) = 0$ dilakukan secara lelaran atau iteratif. Secara umum, semua metode pencarian akar dapat dikelompokkan ke dalam dua golongan besar, yaitu metode tertutup dan metode terbuka. Metode yang termasuk ke dalam golongan metode tertutup mencari akar di dalam selang tertutup $[a, b]$ yang sudah dipastikan memiliki minimal satu buah akar di dalamnya, sehingga lelarannya akan selalu konvergen menuju ke akarnya. Sedangkan untuk metode terbuka tidak diperlukan $[a, b]$ yang mengandung akar. Yang diperlukan adalah tebakan awal akar, lalu, digunakan prosedur lelaran untuk menghitung hampiran akar yang baru. Hampiran akar yang baru tersebut mungkin akan mendekati akar sejati (konvergen), atau mungkin juga menjauhinya (divergen).

2.2 Metode Newton-Raphson

Metode Newton-Raphson merupakan metode untuk pencarian akar-akar pada persamaan nirlanjar yang tergolong dalam metode terbuka. Metode ini paling terkenal dan paling banyak dipakai dalam terapan sains dan rekayasa karena kemampuan konvergensinya yang paling cepat di antara metode lainnya.

Penurunan rumus Netwon-Raphson dapat dilakukan secara geometri.



Gambar 2.1 Tafsiran geometri metode Newton-Raphson

Dari gambar di atas, gradien garis singgung di x_r adalah

$$m = f'(x_r) = \frac{\Delta y}{\Delta x} = \frac{f(x_r) - 0}{x_r - x_{r+1}}$$

atau

$$f'(x_r) = \frac{f(x_r)}{x_r - x_{r+1}}$$

Sehingga prosedur lelaran metode Netwon-Raphson adalah

$$x_r = x_{r+1} \frac{f(x_r)}{f'(x_r)}, \qquad f'(x_r) \neq 0$$

Kondisi berhenti lelaran Newton-Raphson adalah bila

$$|x_{r+1} - x_r| < \varepsilon$$

atau bila menggunakan galat relative hampiran

$$\left|\frac{x_{r+1} - x_r}{x_{r+1}}\right| < \delta$$

dengan ε dan δ merupakan toleransi galat yang diinginkan.

Adapun *pseudocode* untuk melakukan lelaran menggunakan metode Newton-Raphson adalah sebagai berikut.

```
procedure Newton_Raphson(x:real);
{ Mencari akar persamaan f(x) = 0 dengan metode Newton-Raphson
  K.Awal : x adalah tebakan awal akar, nilainya sudah terdefinisi
  K.Akhir: akar persamaan tercetak di layar
}
const
  epsilon = 0.000001;
var
    x_sebelumnya: real;

    function f(x:real):real;
    { mengembalikan nilai f(x). Definisi f(x) bergantung pada persoalan }

    function f_aksen(x:real):real;
    { mengembalikan nilai f'(x). Definisi f'(x) bergantung
     pada persoalan }

begin
  repeat
     x_sebelumnya:=x;
     x:=x - f(x)/f_aksen(x);
  until (ABS(x-x_sebelumnya) < epsilon)

  { x adalah hampiran akar persamaan }
   write('Hampiran akar x = ', x:10:6);

end;
```

3 Solusi

Persamaan

$$c = \frac{1}{a}$$

bisa ditulis sebagai persamaan

$$f(c) = ac - 1 = 0$$

Dari persamaan tersebut, nilai $1/a$ dapat didapatkan dengan cara mencari nilai akar dari persamaan tersebut tanpa menggunakan pembagian. Hal ini dapat dengan mudah diselesaikan dengan menggunakan metode Newton-Raphson.

Metode Newton-Raphson menyelesaikan persamaan nirlanjar $f(c) = 0$ secara iteratif dengan rumus

$$c_{i+1} = c_i - \frac{f(c_i)}{f'(c_i)}$$

yang dalam hal ini c_{i+1} adalah nilai pendekatan baru dari akar persamaan $f(c) = 0$, dan c_i adalah nilai pendekatan sebelumnya dari akar persamaan $f(c) = 0$.

Untuk mendapatkan nilai invers dari bilangan a, perlu didapatkan fungsi yang sesuai. Dengan menggunakan persamaan

$$f(c) = ac - 1 = 0$$

didapatkan

$$f'(c) = a$$

dan dengan menggunakan rumus metode Newton-Raphson didapatkan

$$c_{i+1} = c_i - \frac{ac_i - 1}{a}$$

$$c_{i+1} = \frac{1}{a}$$

Akan tetapi rumus tersebut tidak bisa digunakan karena terdapat pembagian di dalamnya. Oleh sebab itu rumus awalnya diutak-atik lagi sehingga didapatkan

$$f(c) = a - \frac{1}{c} = 0$$

$$f'(c) = \frac{1}{c^2}$$

sehingga jika dimasukkan ke dalam rumus Newton-Raphson memberikan

$$c_{i+1} = c_i - \frac{a - \frac{1}{c_i}}{\frac{1}{c_i^2}}$$

$$= c_i - c_i^2 \left(a - \frac{1}{c_i}\right)$$

$$= c_i(2 - c_i a)$$

Dengan menggunakan nilai awal c_0 untuk mendapatkan inverse dari bilangan a, bisa didapatkan nilai pendekatan baru di setiap lelarannya dengan menggunakan rumus di atas. Setiap lelaran melakukan dua kali operasi perkalian dan satu kali operasi pengurangan. Meskipun begitu, banyaknya iterasi yang dilakukan untuk mendapatkan nilai invers dari a sangat bergantung pada nilai tebakan pendekatan awalnya. Semakin mendekati nilai sejatinya, semakin sedikit jumlah iterasi yang akan dilakukan.

Nilai tebakan awal dipilih berdasarkan rentang $0 < c_0 < \frac{2}{a}$, karena domain kekonvergenan fungsi $c_{i+1} = c_i(2 - c_i a)$ terletak pada rentang tersebut.

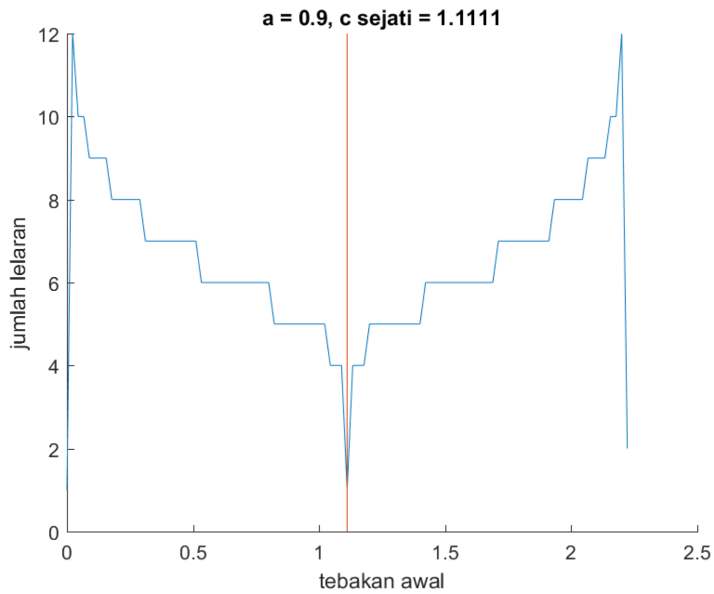
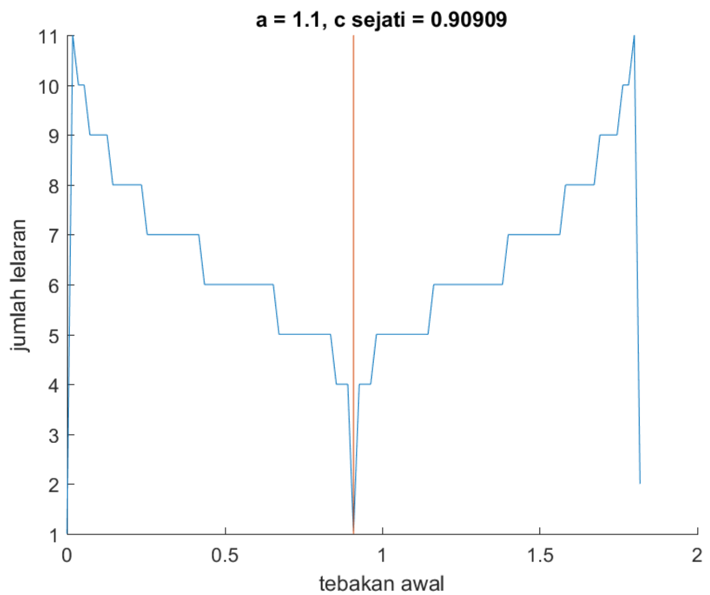
4 Pengujian dan Analisis

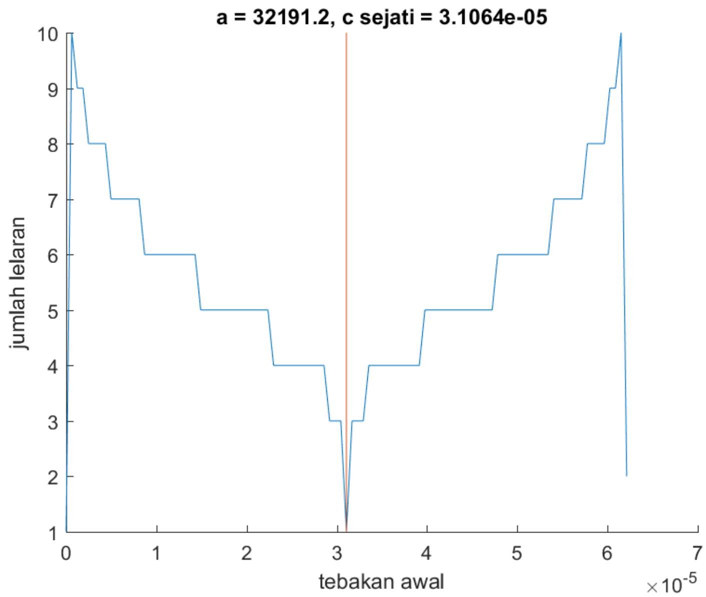
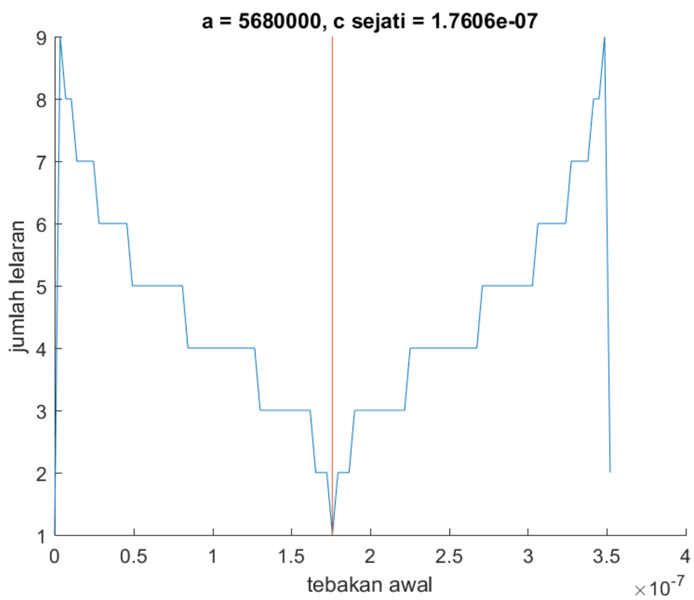
Berdasarkan hipotesis sebelumnya yang mengatakan bahwa banyaknya iterasi yang dilakukan untuk mendapatkan nilai invers dari a sangat bergantung pada nilai tebakan pendekatan awalnya, maka akan dilakukan beberapa pengujian untuk membuktikan hipotesis tersebut. Selain itu, jumlah lelaran juga dipengaruhi oleh nilai toleransi galat (ε).

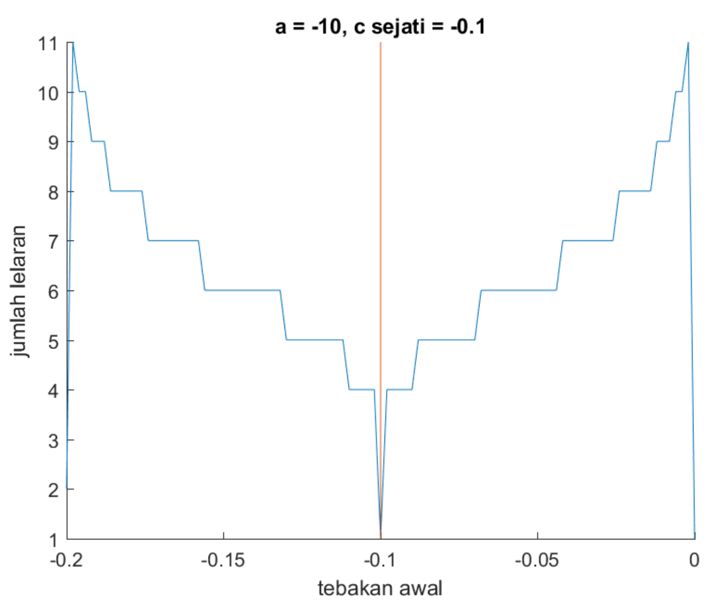
Pengujian akan dilakukan pada dua studi kasus, yaitu:

1. Pengujian titik tebakan awal, dengan nilai-nilai titik di sekitar solusi sejati dengan rentang $0 < c_0 < \frac{2}{a}$. Pengujian dilakukan untuk beberapa nilai a. Nilai toleransi galatnya ditentukan bernilai 0.000000001.
2. Pengujian nilai toleransi galat (ε). Pengujian dilakukan untuk nilai $a = 33$.

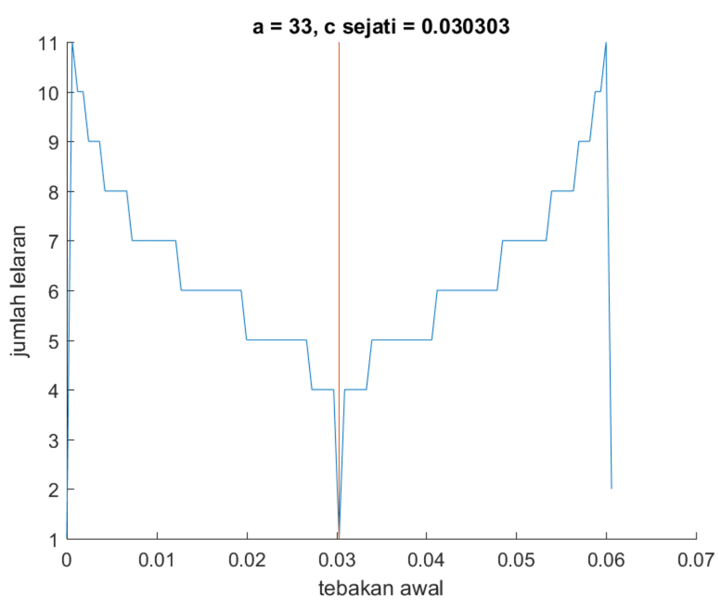
Tabel 4.1 Tabel hasil pengujian kasus pertama

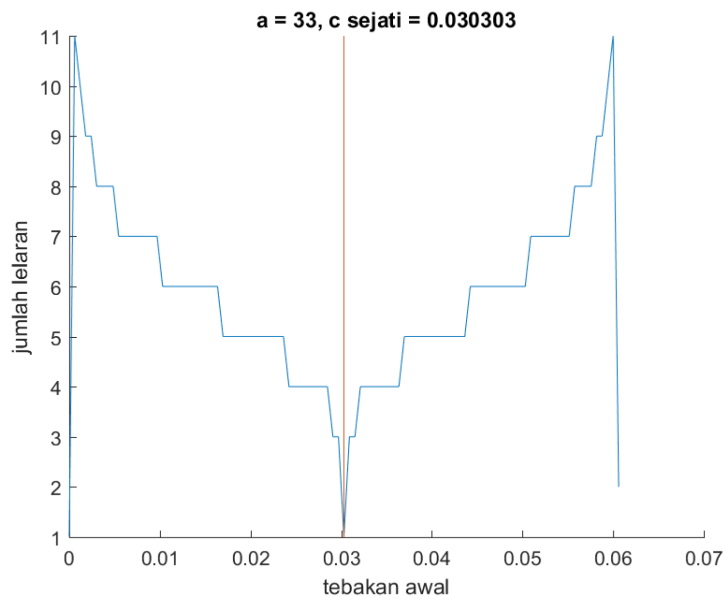
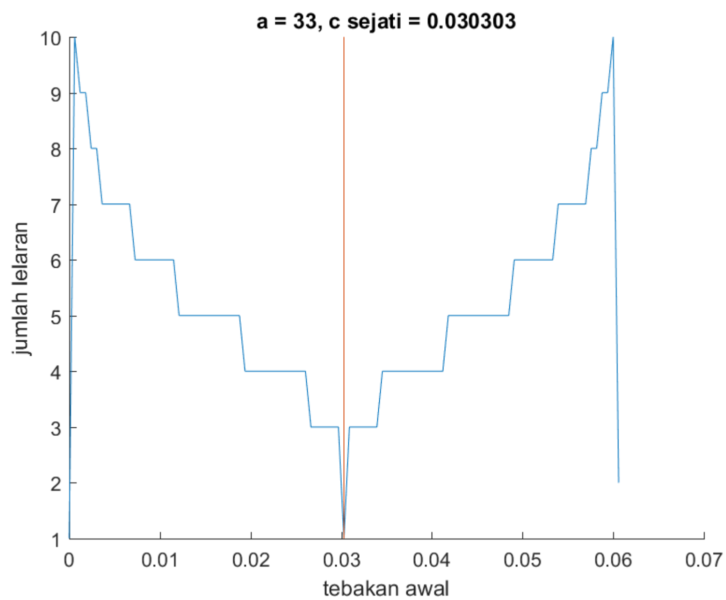
Nilai a	Grafik	Jumlah lelaran maksimal
0,9		12
1,1		11

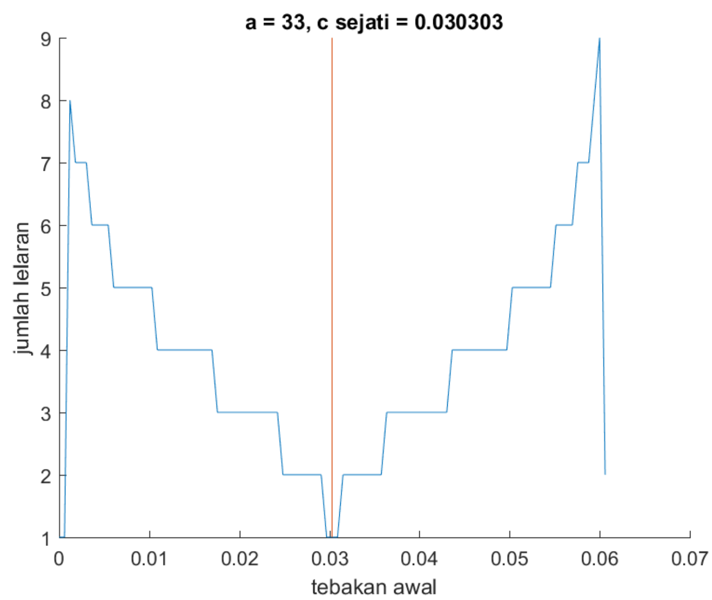
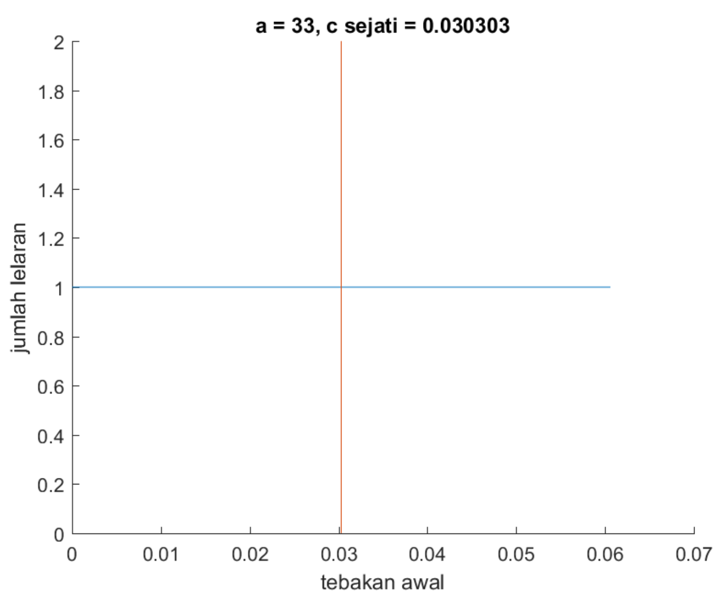
32.191,2		10
5.680.000		9

-10		11

Tabel 4.2 Tabel hasil pengujian kasus kedua

Nilai toleransi galat	Grafik	Jumlah lelaran maksimal
0,000000001		11

0,0000001		11
0,00001		10

0,001		9
0,1		1

Dari hasil pengujian dengan dua kasus di atas, bisa didapatkan beberapa pemahaman sebagai berikut.

Dilihat dari pengujian kasus pertama untuk melihat apakah titik tebakan awal berpengaruh pada jumlah lelaran yang akan dilakukan. Bisa dilihat dari semua hasil pada pengujian kasus pertama bahwa semakin dekat titik awal dengan nilai sejatinya semakin sedikit jumlah lelaran yang harus dilakukan. Titik tebakan awal harus berada pada rentang $0 < c_0 < \frac{2}{a}$ untuk memastikan bahwa lelarannya konvergen. Semakin dekat titik tebakan awal dengan batas ujung rentang, semakin banyak pula jumlah lelaran yang akan dilakukan.

Dapat juga diperhatikan bahwa semakin besar nilai a semakin kecil pula jumlah lelaran maksimal yang dilakukan. Hal ini terjadi dikarenakan nilai invers dari a adalah $c = \frac{1}{a}$, sehingga semakin besar nilainya maka semakin kecil nilai inversnya. Dengan demikian jumlah lelaran maksimal yang dilakukan dipengaruhi oleh nilai toleransi galat yang dipilih, dan dibuktikan oleh hasil pengujian dengan menggunakan kasus kedua. Dengan semakin besar nilai toleransi galat yang dipilih, semakin kecil juga tingkat presisi dari pendekatan yang dihasilkan, sehingga jumlah lelaran minimal yang dilakukan juga akan berkurang.

Jumlah lelaran minimal yang selain satu, berada pada rentang dua sampai empat lelaran sehingga sudah bisa memberikan hasil yang memuaskan karena hasil pembagian bisa lebih cepat didapatkan dibandingkan menggunakan unit pembagian. Hal ini dicapai ketika titik tebakan awal berada pada rentang $(c_{sejati} - 0.01) < c_0 < (c_{sejati} + 0.01)$. Untuk mendapatkan nilai tebakan awal yang sudah hampir mendekati nilai sejatinya ini bisa menggunakan sebuah *look-up table*, berdasarkan riset yang dilakukan oleh (Wong dkk., 1994).

5 Kesimpulan

Kesimpulan yang didapatkan setelah melakukan pengujian yang tidak terlalu banyak adalah (1) banyaknya iterasi yang dilakukan untuk mendapatkan nilai invers dari a sangat bergantung pada nilai tebakan pendekatan awalnya dan (2) jumlah lelaran juga dipengaruhi oleh nilai toleransi galat (ε). Dengan semakin besar nilai toleransi galat yang dipilih, semakin kecil juga tingkat presisi dari pendekatan yang dihasilkan, sehingga jumlah lelaran minimal yang dilakukan juga akan berkurang.

6 Referensi

[1] Munir, Rinaldi. 2003. Metode Numerik. Bandung: Penerbit Informatika.

[2] https://www.mathworks.com/help/simulink/slref/hdlreciprocal.html

[3] Wong, D., dan Flynn, M. (1992): Fast Division Using Accurate Quotient Approximations to Reduce the Number of Iterations, *IEEE Transactions on Computers*, **41**(8), 981–995, diperoleh melalui situs internet: https://doi.org/10.1109/12.156541.

7 Lampiran

```
function [ c, n ] = FindReciprocal( a, x )
% mencari reciprocal dari a dengan menggunakan metode
Newton-Raphson,
% dengan x merupakan tebakan awal

    % toleransi galat yang diinginkan
    epsilon1 = 0.000000001;
    epsilon2 = 0.000000001;
    nmax = 50;

    n = 0;
    berhenti = false;
    x_prev = 0;
    while 1
        if (abs(1/x^2) < epsilon2)
            berhenti = true;
        else
            x_prev = x;
            x = x * (2 - x * a);
            n = n + 1;
        end

        if (abs(x-x_prev) < epsilon1 || berhenti || n
> nmax)
            break
        end
    end

    if (berhenti)
        c = Inf;
    elseif n > nmax
        c = Inf;
    else
        c = x;
    end
end
```

Gambar 7.1 *source code* untuk mencari invers sebuah bilangan dengan menggunakan metode Newton-Raphson

```
addpath(genpath('Functions'));

a = linspace(1, 100, 101);
for j = 1:101
    c_sejati = 1/a(j);
    x = linspace(0, 2/a(j), 101);

    c = zeros(1, 101);
    n = zeros(1, 101);
    for i = 1: 101
        [c(i), n(i)] = FindReciprocal(a(j), x(i));
    end

    h = figure; set(h, 'Visible', 'off');
    hold on
    plot(x, n)
    plot([c_sejati c_sejati], get(gca,'ylim'))
    hold off
    title(['a = ', num2str(a(j)), ', c sejati = ',
num2str(c_sejati)]);
    xlabel('tebakan awal');
    ylabel('jumlah lelaran');

    saveas(h, ['plots/', num2str(j)], 'png');
end
```

Gambar 7.2 *Source code* untuk melakukan pengujian