Perbandingan Kecepatan Komputasi Beberapa Algoritma Solusi Persamaan Nirlanjar

Bernardino Madaharsa Dito Adiwidya - 13507089[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]if17089@students.if.itb.ac.id

***Abstrak*—Suatu persamaan nirlanjar dapat diselesaikan dengan berbagai cara. Cara yang dapat menyelesaikannya dengan lebih sederhana meskipun hasilnya hanya mendekati hasil sebenarnya yaitu dengan metode numerik. Berbagai metode telah dikembangkan untuk menemukan akar dari persamaan nirlanjar. Dalam makalah ini digunakan lima macam metode yaitu Bagi Dua, Regula Falsi, Newton-Raphson, Secant, dan Müller. Kecepatan penghitungan untuk setiap metode tersebut bervariasi, tergantung jenis persamaan yang dihitung. Tidak ada metode yang dinyatakan paling cepat dibandingkan yang lain dalam melakukan penghitungan untuk semua jenis persamaan.**

***Kata kunci*—akar, kecepatan, metode numerik, persamaan nirlanjar.**

I. PENDAHULUAN

Pada saat ini, alat yang digunakan untuk melakukan komputasi sudah sangat banyak digunakan. Alat tersebut dapat berupa sempoa hingga superkomputer. Bahkan pengguna tidak menyadari jika telah menjalankan alat yang terus-menerus melakukan perhitungan seperti pada telepon selular.

Perhitungan rumit selalu dapat menyertai alat-alat tersebut. Perhitungan tersebut bisa saja menjadi sulit diimplementasi secara sempurna pada alat-alat digital. Perhitungan tersebut biasanya merupakan persoalan matematika yang diterapkan dalam berbagai bidang. Karena diperlukan pencarian solusi secara cepat dan praktis, maka diperlukan suatu cara untuk menemukan solusi tersebut meskipun hasilnya hanya mendekati solusi sebenarnya. Solusi sebenarnya hanya dapat diperoleh dengan metode analitik. Untuk menemukan solusi yang mendekati solusi sebenarnya, digunakanlah metode numerik.

Ada berbagai persoalan matematika yang tidak mudah ditemukan solusi sebenarnya. Pada polinom derajat dua sebenarnya telah ada rumus *abc* yang telah dikenal, $x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})/(2a)$. Akan tetapi, untuk polinom derajat lebih dari dua, diperlukan suatu penghitungan khusus. Semakin besar derajat polinom, semakin sulit dilakukan pencarian akarnya. Ada alternatif cara untuk melakukan pencarian tersebut, misalnya dengan metode pembagian sintetis Horner atau menggunakan grafik. Dengan metode pembagian sintetis Horner, perlu dilakukan percobaan perhitungan dengan berbagai bilangan. Dengan grafik dilakukan penghitungan fungsi untuk berbagai nilai. Akan tetapi, persamaan yang digunakan hanya dapat digunakan untuk persamaan dengan maksimal tiga peubah karena penggambaran grafik terbatas pada dimensi tiga saja, tidak mungkin lebih dari tiga [1].

Dengan menggunakan metode numerik, suatu solusi suatu persamaan matematika dapat ditemukan. Solusi tersebut hanya menghampiri atau mendekati solusi sejati saja sehingga solusi tersebut dinamakan dengan solusi hampiran atau solusi pendekatan [1]. Selisih antara solusi sebenarnya dengan solusi hampiran disebut dengan galat (*error*). Oleh karena itu, orang-orang terus mengembangkan metode yang lebih baik untuk meminimalkan galat tersebut.

Pada makalah ini, metode numerik digunakan untuk melakukan komputasi untuk menemukan solusi persamaan nirlanjar. Saat ini telah ada berbagai macam metode untuk menemukan solusi tersebut. Masing-masing metode tersebut memiliki kecepatan komputasi yang berbeda-beda. Oleh karena itu, dalam makalah ini dibahas mengenai perbandingan kecepatan komputasi untuk menemukan solusi beberapa persamaan. Perbandingan ini mengimplementasi beberapa metode yang akan dibandingkan dalam suatu aplikasi *desktop*.

II. TEORI SINGKAT METODE PENCARI SOLUSI PERSAMAAN NIRLANJAR

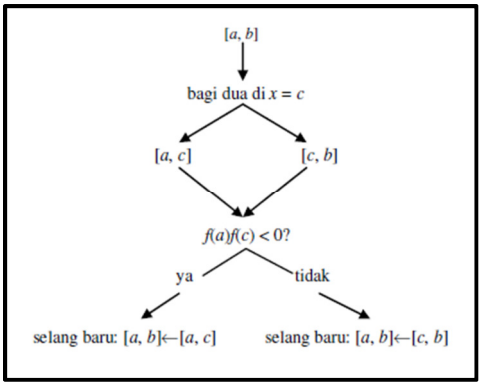
Dalam metode numerik, pencarian akar suatu persamaan nirlanjar dilakukan dengan iterasi. Secara umum ada dua golongan besar metode ini, yaitu metode tertutup dan terbuka. Metode tertutup menggunakan selang [*a*, *b*] yang telah dipastikan memiliki akar di antaranya. Iterasi yang dilakukan selalu konvergen ke akar sehingga metode ini disebut juga dengan metode konvergen. Metode terbuka tidak memerlukan selang, melainkan tebakan angka. Dengan metode terbuka, ada kemungkinan hampiran angka yang ditemukan mendekati akar sejati atau malahan menjauhinya. Oleh karena itu, metode ini tidak selalu konvergen ataupun divergen.

Dalam makalah ini, digunakan lima metode yang digunakan untuk mencari solusi persamaan nirlanjar.

Kelima metode tersebut antara lain metode Bagi Dua, Regula Falsi, Newton-Raphson, Secant, dan Müller. Metode Bagi Dua dan Regula Falsi tergolong metode tertutup, sedangkan metode Newton-Raphson, Secant, dan Müller tergolong metode terbuka.

A. Metode Bagi Dua

Metode ini memerlukan selang [*a*, *b*] yang di antaranya diyakini terdapat solusi. Setelah menentukan selang tersebut, ditentukan suatu nilai *c* yang merupakan titik tengah dari selang tersebut dan diperoleh dua selang: [*a*, *c*] dan [*c*, *b*]. Selang yang diambil untuk iterasi berikutnya adalah selang yang memiliki akar, tergantung nilai $f(a)f(c) < 0$ atau $f(c)f(b) < 0$.



Gambar 1 Skema Metode Bagi Dua [1]

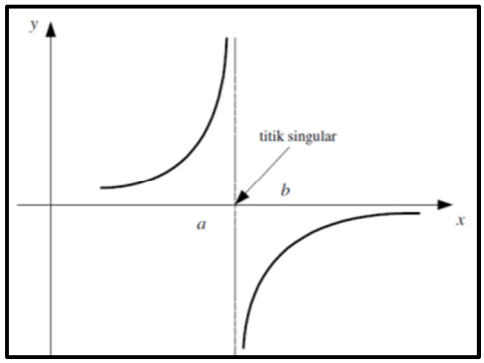
Kondisi pada gambar di atas berhenti jika $|a - b| < \varepsilon$, dengan ε merupakan lebar selang sempit yang diinginkan. Dengan cara di atas, ada kemungkinan nilai *f*(*c*) sudah sama dengan nol. Oleh karena itu diperlukan penanganan khusus untuk hal ini dengan membandingkan jika selisih antara *f*(*c*) dengan 0 kurang dari epsilon mesin. Kondisi berhenti pencarian ini dapat menggunakan galat relatif hampiran di samping berdasarkan ukuran selang, $abs((c_{baru} - c_{lama})/c_{baru}) < \delta$, dengan δ merupakan galat relatif hampiran yang diinginkan.

Berikut ini adalah *pseudocode* (mirip Pascal) untuk metode Bagi Dua.

```
procedure BagiDua(input a,b: real,
                  output c:real)
const epsilon = 0.000001
      epsilonmesin = ...
begin
  repeat
    c ← (a+b)/2
    if abs(f(c)) < epsilonmesin then
      begin
        a ← c
        b ← c
      end
    else
      if f(a)*f(c) < 0 then b ← c
      else a ← c
  until abs(a-b) < epsilon
end
```

Metode ini memiliki beberapa kasus yang mungkin terjadi dalam penggunaannya dan memerlukan penanganan khusus untuk mengatasinya, antara lain:

1. Jumlah akar lebih dari satu
   Kasus ini terjadi bila dalam selang [*a*, *b*] terdapat lebih dari satu akar. Metode ini hanya mampu menemukan satu akar saja dalam selang tersebut.
2. Akar ganda
   Contoh kasusnya adalah untuk persamaan $f(x) = (x-3)^2$. Akar yang diperoleh ada dua, yaitu sama-sama pada *x* = 3.
3. Singularitas

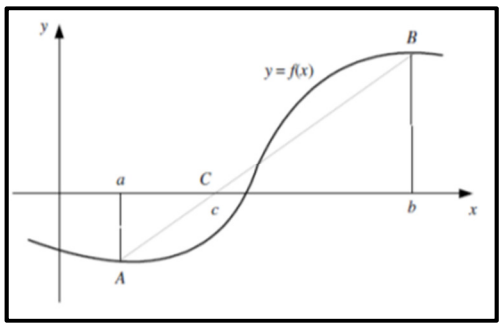


Gambar 2 Singularitas [1]

Jika terdapat kasus ini, proses metode Bagi Dua terus berjalan. Titik singular dianggap sebagai akar karena iterasi cenderung konvergen. Untuk mengatasinya, setiap kali melakukan iterasi baru dilakukan pengecekan kekonvergenan |*f*(*b*) - *f*(*a*)|.

B. Metode Regula Falsi

Metode ini dikembangkan dari metode Bagi Dua akibat kecepatan konvergensi yang lambat. Kecepatan konvergensi dapat ditingkatkan jika nilai *f*(*a*) dan *f*(*b*) juga diperhitungkan. Jika *f*(*a*) lebih dekat ke nol daripada *f*(*b*), tentu akar lebih dekat ke *x* = *a* daripada *x* = *b*.

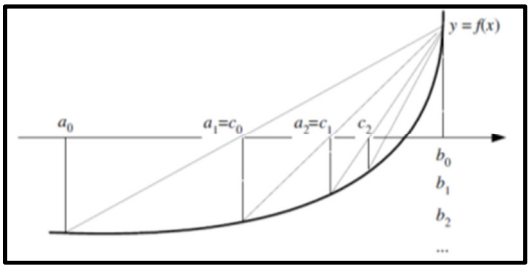


Gambar 3 Skema Metode Regula Falsi [1]

Berdasarkan gambar di atas, nilai *c* dapat diperoleh dengan membandingkan gradian garis AB dengan BC.

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

Meskipun telah disebutkan jika metode ini lebih cepat daripada metode Bagi Dua, ada kemungkinan metode ini justru malah menjadi lebih lambat. Kasus ini terjadi jika kurva fungsinya cekung pada selang [*a*, *b*] yang mengakibatkan garis potongnya selalu berada di atas atau di bawah kurva. Hal ini memunculkan kejadian ada titik ujung selang yang tidak berubah, yang disebut dengan titik mandek.



Gambar 4 Kurva Fungsi Cekung [1]

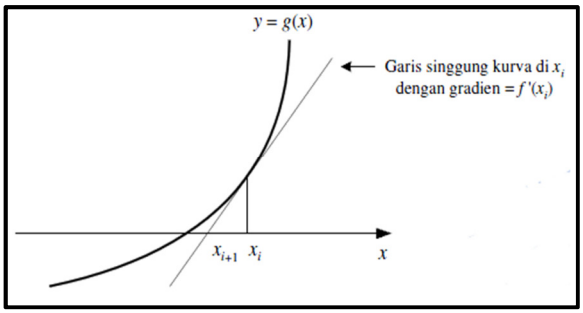
Untuk mengatasi hal tersebut, kondisi berhenti untuk *loop* harus ditambah dengan melakukan pengecekan nilai *f*(*c*) yang sudah sangat kecil sehingga mendekati nol. Berikut ini adalah *pseudocode* (mirip Pascal) untuk metode Regula Falsi yang telah diperbaiki.

```
procedure regula_falsi(input a, b: real,
                       output c: real)
const epsilon = 0.00001
      delta = 0.000001
      epsilonmesin = ...
begin
repeat
  FA ← f(a)
  FB ← f(b)
  mandekkiri ← 1
  mandekkanan ← 1
  repeat
    c ← b - (FB*(b-a)/(FB-FA))
    if abs(f(c)) < epsilonmesin then
      begin
        a ← c
        b ← c
      end
    else
      begin
        if f(a)*f(c) < 0 then
          begin
            b ← c
            FB ← f(c)
            mandekkiri ← mandekkiri + 1
            mandekkanan ← 0
            if mandekkiri > 1 then
              FA ← FA/2
          end
        else
          begin
            a ← c
            FB ← f(c)
            mandekkanan ← mandekkanan + 1
            mandekkiri ← 0
            if mandekkanan > 1 then
              FB ← FB/2
          end
      end
  until (abs(a-b)<epsilon) or
        (abs(f(c))<delta)
end
```

C. Metode Newton-Raphson

Metode ini paling terkenal dan banyak dipakai dalam terapan sains dan rekayasa. Metode ini paling disukai karena konvergensinya paling cepat [1]. Ada dua pendekatan dalam penurunan rumus metode ini yaitu:

1. Tafsiran Geometri



Gambar 5 Tafsiran Geometri [1]

Berdasarkan gambar di atas, gradien garis singgung di x_i adalah $m = f'(x_i) = \frac{\Delta y}{\Delta x} = \frac{f(x_i)-0}{x_i - x_{i+1}}$ sehingga diperoleh prosedur lelaran metode ini adalah $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ dengan $f'(x_i) \neq 0$.

2. Deret Taylor

Pertama-tama, $f(x_r)$ diuraikan di sekitar x_r ke dalam deret Taylor: $f(x_{r+1}) \approx f(x_r) + (x_{r+1} - x_r)f'(x_r) + \frac{(x_{r+1}-x_r)^2}{2} f''(t), x_r < t < x_{r+1}$. Jika dipotong sampai suku orde kedua saja, hasilnya menjadi $f(x_{r+1}) \approx f(x_r) + (x_{r+1} - x_r)f'(x_r)$. Karena persoalannya mencari akar, maka $f(x_{r+1}) = 0$ sehingga diperoleh $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ dengan $f'(x_i) \neq 0$.

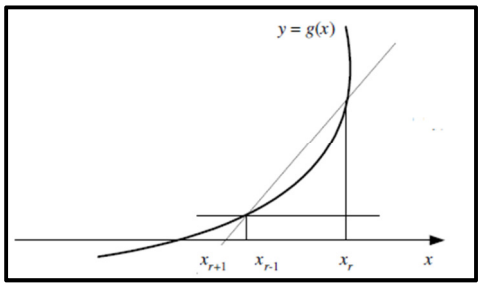
Kondisi berhenti lelaran Newton-Raphson adalah jika $|x_{r+1} - x_r| < \varepsilon$ atau $\left|\frac{x_{r+1}-x_r}{x_{r+1}}\right| < \delta$. Berikut ini adalah *pseudocode* (mirip Pascal) untuk metode ini.

```
procedure Newton_Raphson(
          input/output x:real)
const epsilon = 0.000001
var
  x_sebelumnya, fa: real
  function f(x:real): real
  function f_aksen(x:real): real
  nol: boolean
begin
  nol ← false
  repeat
```

```
    x_sebelumnya ← x
    fa ← f_aksen(x)
    if (fa≠0) then x ← x - f(x)/fa
    else nol ← true
  until (ABS(x-x_sebelumnya)<epsilon)and
        (!nol)
end;
```

D. Metode Secant

Metode ini merupakan modifikasi dari metode Newton-Raphson. Metode ini dibuat dengan menggantikan penghitungan turunan fungsi.



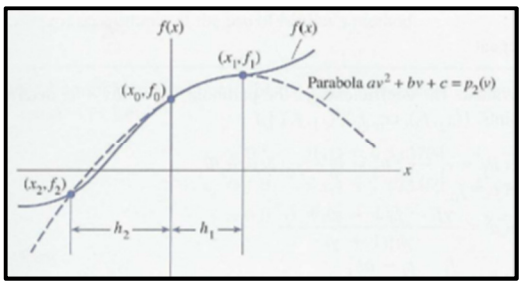
Gambar 6 Metode Secant [1]

Dari gambar di atas diperoleh nilai $f'(x_r) = \frac{\Delta y}{\Delta x} = \frac{f(x_r)-f(x_{r-1})}{x_r-x_{r-1}}$. Nilai tersebut disubtitusikan dalam rumus Newton-Raphson sehingga diperoleh nilai $x_{r+1} = x_r - \frac{f(x_r)(x_r-x_{r-1})}{f(x_r)-f(x_{r-1})}$. Berikut ini *pseudocode* (mirip Pascal) untuk metode ini.

```
procedure Secant(input x0, x1:real,
                 output x:real)
const epsilon = 0.000001
var
  x_sebelumnya: real
  function f(x:real):real
begin
  repeat
    x_sebelumnya ← x1
    x ← x-(f(x1)*(x1 - x0)/(f(x1)-f(x0)))
    x0 ← x1
    x1 ← x
  until (ABS(x-x_sebelumnya) < epsilon)
end;
```

E. Metode Müller

Metode ini mendekati fungsi dengan polinomial kuadratik yang lebih cocok dengan fungsi tersebut daripada garis lurus. Cara ini secara signifikan meningkatkan tingkat konvergensi di atas interpolasi linear [2].



Gambar 7 Metode Müller [2]

Dari gambar di atas, polinomial dibentuk untuk mencocokkan tiga buah titik, yaitu pada titik x_0, x_1, dan x_2 dengan titik x_0 berada di tengah kedua titik lainnya. Persamaan kuadratik yang melewati ketiga titik tersebut berupa persamaan $av^2 + bv + c$. Kemudian, dilakukan langkah-langkah penghitungan sebagai berikut.

1. $v = x - x_0$
2. $h_1 = x_1 - x_0$
3. $h_2 = x_0 - x_2$
4. Misal $v = 0 \rightarrow a.0^2 + b.0 + c = f_0$
5. Misal $v = h_1 \rightarrow ah_1^2 + bh_1 + c = f_1$
6. Misal $v = -h_2 \rightarrow ah_2^2 - bh_2 + c = f_2$
7. Dari nomor 4 diperoleh $c = f_0$
8. Dengan nilai $h_2/h_1 = \gamma$, maka dapat diperoleh nilai $a = \frac{\gamma f_1 - f_0(1+\gamma)+f_2}{\gamma h_1^2(1+\gamma)}$ dan $b = \frac{f_1-f_0-ah_1^2}{h_1}$.
9. Setelah menemukan nilai a, b, dan c, persamaan $av^2 + bv + c$ dapat diselesaikan dengan rumus $v_{1,2} = \frac{2c}{-b\pm\sqrt{b^2-4ac}}$. Karena $v = x - x_0$, akar yang dapat ditemukan adalah $x_0 - \frac{2c}{b\pm\sqrt{b^2-4ac}}$

III. PENGUJIAN

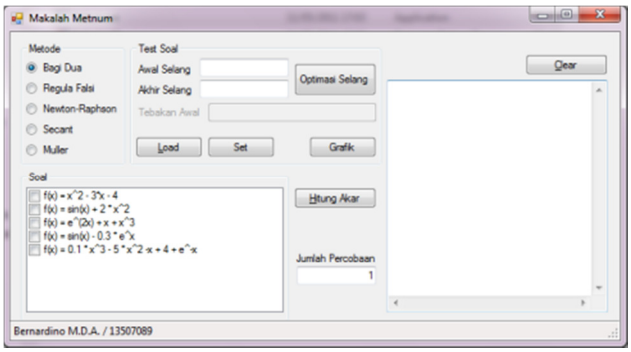
Pengujian dilakukan dengan membuat sebuah aplikasi sederhana yang mengimplementasi kelima metode tersebut. Aplikasi tersebut merupakan aplikasi *desktop*. Aplikasi akan menghitung solusi dari beberapa soal yang telah disiapkan. Setiap kali melakukan perhitungan pada setiap soal, aplikasi akan mengukur waktu yang diperlukan untuk komputasi tersebut.

A. Spesifikasi Tempat Pengujian

Pengujian ini dilakukan pada *notebook* dengan prosesor AMD Turion X2 Dual-Core Mobile RM-75 2,2Ghz. Memori yang digunakan 3072MB RAM. Sistem operasi yang digunakan adalah Windows 7 Ultimate 32-bit.

B. Spesifikasi Aplikasi

Aplikasi *desktop* ini dibuat untuk sistem operasi Windows. Aplikasi dibuat dengan menggunakan Visual Studio 2008 dengan .NET Framework 3.5.



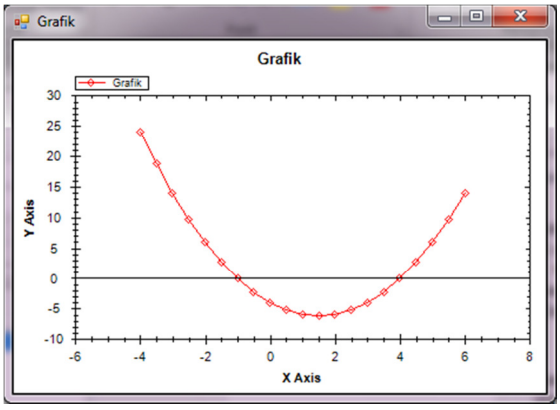
Gambar 8 Screenshot Aplikasi

Aplikasi tersebut memungkinkan percobaan penghitungan lebih dari satu kali sehingga memudahkan pengukuran waktu. Selain itu, terdapat tombol untuk menghasilkan grafik dari persamaan yang dipilih. *Library* grafik yang digunakan adalah ZedGraph.

C. Spesifikasi Soal

Untuk pengujian ini, digunakan lima buah persamaan yang akan dicari akarnya. Untuk keperluan penghitungan oleh setiap metode, diperlukan nilai selang dan tebakan. Nilai selang diperlukan oleh metode Bagi Dua, Regula Falsi, dan Secant. Nilai tebakan diperlukan oleh metode Newton-Raphson dan Müller. Nilai-nilai tersebut diperoleh dari perkiraan posisi akar dari grafik untuk setiap soal. Berikut ini soal, nilai-nilai yang diperlukan, dan grafik masing-masing soal.

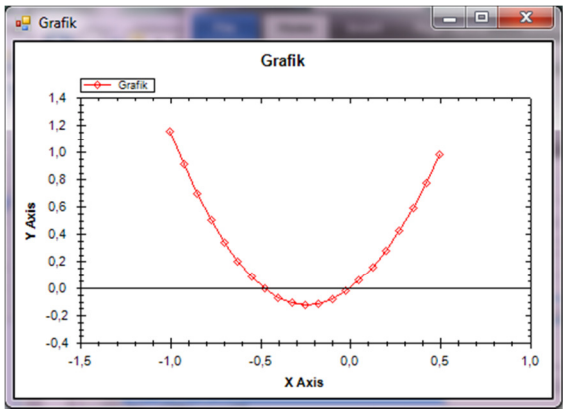
1. $f(x) = x^2 - 3x - 4$
   $f'(x) = 2x - 3$



Gambar 9 Grafik Fungsi Pertama

   Selang : [3, 5]
   Tebakan : 3,5; 3,8; 4,5

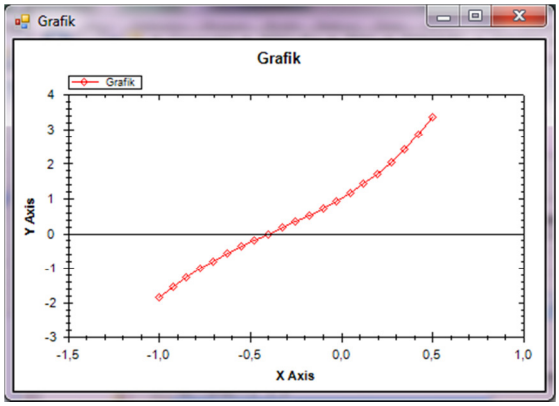
2. $f(x) = \sin x + 2x^2$
   $f'(x) = \cos x + 4x$



Gambar 10 Grafik Fungsi Kedua

   Selang : [-0,2; 0,2]
   Tebakan : -0,1; -0,08; 0,1

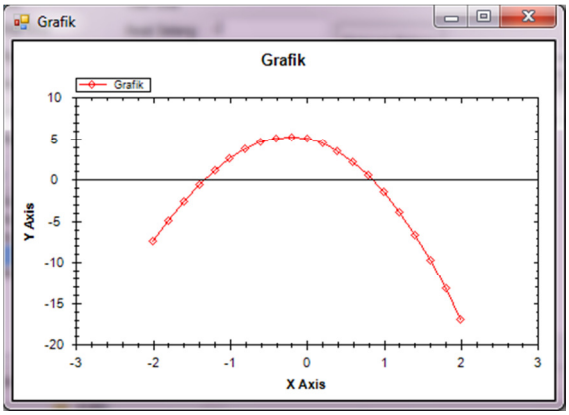
3. $f(x) = e^{2x} + x + x^3$
   $f'(x) = 2e^{2x} + 1 + 3x^2$



Gambar 11 Grafik Fungsi Ketiga

   Selang : [-1, 1]
   Tebakan : -0,1; -0,2; -0,6

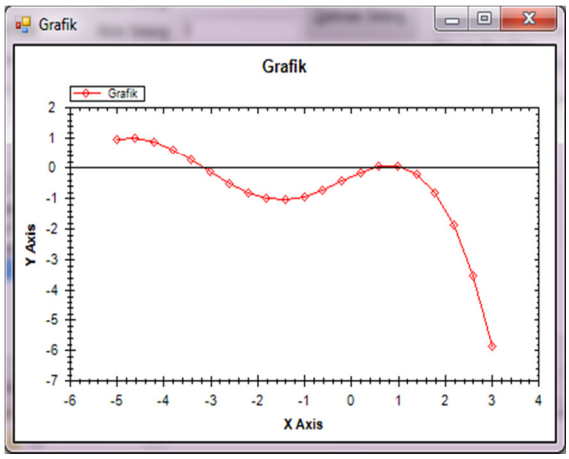
4. $f(x) = \sin x - 0.3e^x$
   $f'(x) = \cos x - 0.3e^x$



Gambar 12 Grafik Fungsi Keempat

   Selang : [-1,5; -0,5]
   Tebakan : -1,1; -1,2; -1,6

5. $f(x) = 0.1x^3 - 5x^2 - x + 4 + e^{-x}$
   $f'(x) = 0.3x^2 - 10x - 1 - e^{-x}$



Gambar 13 Grafik Fungsi Kelima

Selang : [-4, -2]
Tebakan : -2,5; -2,8; -4

D. Hasil Penghitungan Solusi

Untuk setiap soal, solusi setiap metode yang diperoleh sebagai berikut.

1. Soal 1
   Bagi Dua : 4
   Regula Falsi : 3,99999999999999
   Newton-Raphson : 4
   Secant : 4
   Müller : 4
2. Soal 2
   Bagi Dua : 0
   Regula Falsi : $3{,}2223378289677.10^{-13}$
   Newton-Raphson : 0
   Secant : $3{,}02214519460815.10^{-31}$
   Müller : $-1{,}25346842395998.10^{-25}$
3. Soal 3
   Bagi Dua : -0,393821716308594
   Regula Falsi : -0,393827552889356
   Newton-Raphson : -0,393827552889287
   Secant : -0,393827552889287
   Müller : -0,393827552889287
4. Soal 4
   Bagi Dua : -1,33345794677734
   Regula Falsi : -1,33345184463448
   Newton-Raphson : -1,33345184463448
   Secant : -1,33345184463448
   Müller : -1,33345184463448
5. Soal 5
   Bagi Dua : -3,15439605712891
   Regula Falsi : -3,15439229916108
   Newton-Raphson : -3,15439229916108
   Secant : -3,15439229916108
   Müller : -3,15439229916108

E. Hasil Penghitungan Kecepatan

Dalam makalah ini, untuk memperoleh ketelitian yang lebih baik, jumlah perhitungan setiap soal sebanyak 1.000.000 kali. Berikut ini perbandingan perhitungan setiap soal untuk masing-masing metode.

Tabel 1 Hasil Pengukuran Waktu

Metode	Waktu Hitung (ms) untuk 1.000.000 Soal				
	1	2	3	4	5
Bagi Dua	856	49	218	559	393
Regula Falsi	259	916	39	714	398
Newton-Raphson	808	463	4	178	562
Secant	95	802	346	141	930
Müller	743	565	974	600	172

IV. ANALISIS

Berdasarkan tabel hasil pengukuran waktu perhitungan di atas, diperoleh analisis untuk masing-masing metode untuk setiap soal sebagai berikut.

1. Soal 1: $f(x) = x^2 - 3x - 4$
   Soal ini hanya melibatkan polinomial derajat dua. Metode yang paling cepat untuk menyelesaikannya adalah metode Secant. Metode yang cukup cepat berikutnya adalah Regula Falsi. Ketiga metode lainnya sangat lambat untuk menyelesaikan jenis soal ini.
2. Soal 2: $f(x) = \sin x + 2x^2$
   Soal ini melibatkan persamaan polinomial dengan trigonometri. Metode yang paling cepat adalah metode Bagi Dua. Metode yang cukup cepat menyelesaikannya adalah metode Newton-Raphson dan Müller, sedangkan kedua metode sisanya sangat lambat.
3. Soal 3: $f(x) = e^{2x} + x + x^3$
   Soal ini melibatkan persamaan polinomial derajat tiga dengan eksponensial. Metode paling cepat menyelesaikannya adalah Newton-Raphson. Regula Falsi juga cukup cepat menyelesaikannya. Metode Müller sangat lambat dalam menyelesaikannya/
4. Soal 4: $f(x) = \sin x - 0.3e^x$
   Soal ini melibatkan trigonometri dan eksponensial. Metode yang cepat menyelesaikan jenis soal ini adalah Secant dan Newton-Raphson. Ketiga metode lainnya cukup lambat dalam penyelesaiannya.
5. Soal 5: $f(x) = 0.1x^3 - 5x^2 - x + 4 + e^{-x}$
   Soal ini melibatkan polinomial derajat dua dan tiga beserta eksponensial. Metode yang paling cepat menyelesaikannya adalah metode Müller. metode-metode lainnya lebih lambat. Metode yang paling lambat menyelesaikannya adalah Secant.

V. KESIMPULAN

Berdasarkan hasil percobaan dan analisis di atas, tidak ada satupun metode yang selalu lebih cepat melakukan komputasi daripada metode yang lain. Kecepatan penghitungan penyelesaian akar suatu persamaan tergantung kerumitannya. Persamaan yang hanya mengandung polinomial derajat rendah sebaiknya diselesaikan dengan metode Secant. Jika suatu persamaan mengandung penghitungan eksponen, perhitungan yang cukup cepat dapat dilakukan oleh Newton-Raphson.

REFERENSI

[1] R. Munir, "Metode Numerik untuk Teknik Informatika," Bandung, 1997.

[2] Solving Nonlinear Equations II, URL:http://siber.cankaya.edu.tr/ozdogan/NumericalComputations/week4/week4p.pdf
Waktu akses: 11 Mei 2011 14:41

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, 13 Mei 2011

Bernardino M.D.A. / 13507089