SOLUSI PERSAMAAN NON LINEAR

1. Latar Belakang

Dalam bidang sains dan rekayasa, para ahli ilmu alam dan rekayasawan sering berhadapan dengan persoalan mencari solusi persamaan – lazim disebut akar persamaan (*root of equation*) atau nilai-nilai nol – yang berbentuk $f(x) = 0$. Beberapa persamaan sederhana mudah ditemukan akarnya. Misalnya, $2x + 3 = 0$, pemecahannya adalah dengan memindahkan 3 ke ruas kanan sehimgga menjadi $2x = -3$, dengan demikian solusi atau akarnya adalah $x = -\frac{3}{2}$. Begitu juga persamaan kuadratik seperti $x^2 - 4x - 5 = 0$, akar-akarnya mudah ditemukan dengan cara pemfaktoran menjadi $(x - 5)(x + 1)$, sehingga $x_1 = 5$ dan $x_2 = -1$.

Umumnya persamaan yang kan dipecahkan muncul dalam bentuk non linear yang melibatkan bentuk sinus, cosines, eksponensial, ligaritma, dan fungsi transenden lainnya. Misalnya, akar real terkecil dari $9{,}34 - 21{,}97x + 16{,}3x^3 - 3{,}704x^5 = 0$.

Contoh diatas memperlihatkan bentuk persamaan yang rumit/ kompleks yang tidak dapat dipecahkan secara analitik (seperti persamaan kuadratik pada paragraph awal). Bila metode analitik tidak dapat menyelesaikan persamaan, maka kita masih bias mencari solusinya dengan mengguakan metode numerik.

Berdasarkan latar belakang diatas, akan dijelaskan beberapa metode dalam penyelesaian persamaan non linear.

2. Rumusan Masalah

Berdasarkan latar belakang diatas, adapun rumusan masalah makalah ini sebagai berikut:

1. Apa saja metode pencarian akar?
2. Metode apa saja yang termasuk dalam metode terbuka?
3. Metode apa saja yang termasuk dalam metode tertutup?

4. Bagaimana penyelesaian persamaan yang memiliki akar ganda?
5. Bagaimana penyelesaian persamaan yang memiliki akar-akar polinom?
6. Bagaimana system penyelesaian persamaan non linear?

3. Tujuan

1. Ingin mengetahui pa saja metode pencarian akar
2. Ingin mengetahui metode apa saja yang termasuk dalam metode terbuka
3. Ingin mengetahui metode apa saja yang termasuk dalam metode tertutup
4. Ingin mengetahui penyelesaian persamaan yang memiliki akar ganda
5. Ingin mengetahui penyelesaian persamaan yang memiliki akar-akar polinom
6. Ingin mengetahui sistem penyelesaian persamaan non linear

BAB II
PEMBAHASAN

A. Metode Pencarian Akar

Dalam metode numerik, pencarian akar $f(x) = 0$ dilakukan secara lelaran (iteratif). Sampai saat ini sudah banyak ditemukan metode pencarian akar. Secara umum semua metode pencarian akar tersebut dapat dikelompokkan menjadi 2 golongan besar:

a) Metode Tertutup Atau Metode Pengurung (*bracketing method*)
   Metode yang termasuk ke dalam golongan ini mencari akar didalam selang $[a, b]$ sudah dipastikan berisi minimal satu buah akar, karena itu metode jenis ini selalu berhasil menemukan akar. Dengan kata lain, lelarannya selalu konvergen ke akar, karena itu metde tertutup kadang-kadang dinamakan juga metode konvergen.

b) Metode terbuka
   Berbeda dengan metode tertutup, metode terbuka tidak memerlukan selang $[a, b]$ yang mengandung akar. Yang diperlukan adalah tebakan awal akar, lalu dengan prosedur lelaran kita menggunakannya untuk menghitung hampiran akar yang baru. Pada setiap kali lelaran, hampiran akar yang lama dipakai untuk menghitung hampiran akar yang baru. Mungkin saja hampiran akar yang baru mendekati akar yang sejati (konvergen), atau mungkin juga menjauhinya (divergen). Kerena itu, metode terbuka tidak selalu berhasil menemukan akar, kadang-kadang konvergen, kadangkala divergen.

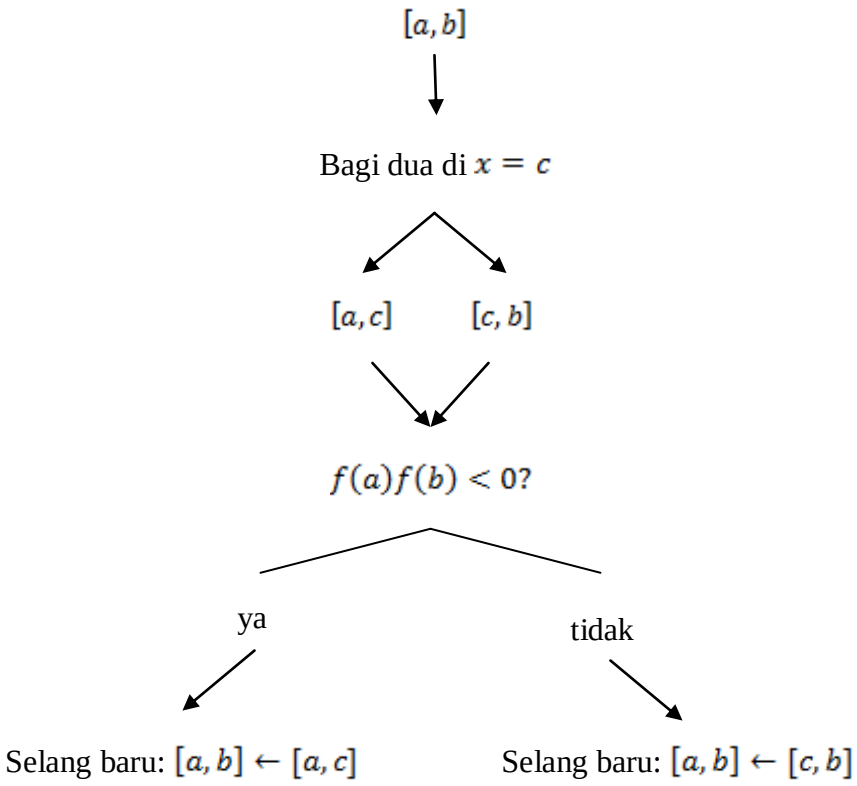
B. Metode Tertutup

Seperti yang telah dijelaskan, metode tertutup memerlukan selang $[a, b]$ yang mengandung akar. Sebagaimana namanya, selang tersebut “mengurung” akar sejati. Strategi yang dipakai adalah mengurangi lebar selang secara sistematis sehingga lebar selang tersebut semakin sempit dan karenanya menuju akar yang benar.

Ada dua metode klasik yang termasuk ke dalam metode tertutup, yaitu metode bagi dua dan metode regula-falsi. Masing-masing metode kita bahas lebih rinci di bawah ini.

1) Metode Bagi Dua atau Metode Bolzano

Misalkan kita telah menentukan selang $[a, b]$ sehingga $f(a)\ f(b) < 0$. Pada setiap kali lelaran, selang$[a, b]$ kita bagi dua di $x = c$, sehingga terdapat dua buah subselang yang berukuran sama yaitu selang $[a, c]$ dan$[c, b]$. Selang yang diambil untuk lelaran berikutnya adalah subselang yang memuat akar, bergantung pada apakah $f(a)\ f(b) < 0$.



Selang yang baru di bagi dua lagi dengan cara yang sama. Begitu seterusnya sampai ukuran selang yang baru sudah sangat kecil. Kondisi berhenti lelaran dapat dipilih salah satu dari tiga kriteria berikut:

1. Lebar selang baru $|a - b| < \varepsilon$ yang dalam hal ini ε adalah nilai toleransi lebar selang yang mengurung akar.
2. Nilai fungsi di hampiran akar: $f(c) = 0$

3. Galat relatif hampiran akar: $|(c_{baru} - c_{lama})/c_{baru}| < \delta$. Yang dalam hal ini δ adalah galat relatif hampiran yang diinginkan.

   Berikut ini program yang berisi algoritma metode bagi dua.

```
procedure BagiDua(a,b: real);
{ Mencari akar f(x)=0 di dalam selang [a,b] dengan metode bagidua
K.Awal : a dan b adalah ujung-ujung selang sehingga
f(a)*f(b) < 0, nilai a dan b sudah terdefinisi.
K.Akhir : Hampiran akar tercetak di layar.
}
const
epsilon1 = 0.000001; {batas lebar selang akhir lelaran}
epsilon2 = 0.00000001; {bilangan yang sangat kecil, mendekati nol}
begin
repeat
c:=(a+b)/2; { titik tengah [a,b]}
if f(a)*f(c) < 0 then
b:=c {selang baru [a,b]=[a,c]}
else
a:=c; {selang baru [a,b]=[c,b]}
until (ABS(a-b)< epsilon1) or (f(c)) < epsilon2);
{ c adalah akar persamaan }
writeln('Hampiran Akar = ', x:10:6);
   End;
```

TEOREMA

Jika $f(x)$ menerus di dalam selang $[a, b]$ dengan $f(a)\ f(b) < 0$ dan $s \in [a, b]$ sehingga $f(s) = 0$ dan $c_r = (a_r + b_r)/2$, maka selalu berlaku dua ketidaksamaan berikut

$$|s - c_r| \leq |b_r - a_r|/2$$

dan

$$|s - c_r| \leq \frac{|b - a|}{2^{r+1}}, r = 0, \quad 1, 2, \ldots$$

Bukti:

Misalkan pada lelaran ke-r kita mendapatkan selang $[a_r, b_r]$ yang panjangnya setengah panjang selang sebelumnya $[a_{r-1}, b_{r-1}]$.

Jadi, $|b_r - a_r| = |b_{r-1} - a_{r-1}|/2$

Jelaslah bahwa

$$|b_1 - a_1| = \frac{|b_0 - a_0|}{2} = \frac{|b-a|}{2}$$

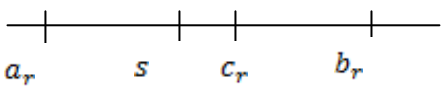
$$|b_2 - a_2| = \frac{|b_1 - a_1|}{2} = \frac{|b-a|}{2^2}$$

$$|b_3 - a_3| = \frac{|b_2 - a_2|}{2} = \frac{|b-a|}{2^3}$$

$$\vdots$$

$$|b_r - a_r| = \frac{|b_{r-1} - a_{r-1}|}{2} = \frac{|b-a|}{2^r}$$

Pada lelaran ke-*r*, posisi c_r (akar himpunan) dan s (akar sejati) adalah seperti diagram berikut



Berdasarkan diagram di atas jelaslah bahwa

$$|s - c_r| \leq |b_r - a_r|/2$$

Selanjutnya,

$$|s - c_r| \leq \frac{|b_r - a_r|}{2} = \frac{1}{2}\frac{|b-a|}{2^r} = \frac{|b-a|}{2^{r+1}}$$

Jadi selisih antara akar sejati dengan akar hampiran tidak pernah lebih dari setengah epsilon.

Dengan mengingat kriteria berhenti adalah $|a - b| < \varepsilon$, maka dari (i) terlihat bahwa

$$|s - c_r| \leq \varepsilon/2$$

sehingga

$$\frac{|b-a|}{2^{r+1}} < \frac{\varepsilon}{2}$$

$$\leftrightarrow \; 2^r > \frac{|b-a|}{\varepsilon}$$

$$\leftrightarrow r \ln 2 > \ln|b-a| - \ln \varepsilon$$

$$\leftrightarrow r > \frac{\ln|b-a| - \ln \varepsilon}{\ln 2}$$

$$\leftrightarrow R > \frac{\ln|b-a| - \ln \varepsilon}{\ln 2}$$

Yang dalam hal ini *R* adalah jumlah lelaran (jumlah pembagian selang) yang dibutuhkan untuk menjamin bahwa c adalah hampiran akar yang memiliki galat kurang dari ε.

Contoh 3.1

Tentukan akar $f(x) = e^x - 5^x$ di dalam selang $[0,1]$ dan $\varepsilon = 0{,}00001$

Tabel lelaran menggunakan Metode Bagidua:

r	a	c	b	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebarnya
0	0,000000	0,500000	1,000000	1,000000	0,398721	-2,281718	$[c, b]$	0,500000
1	0,500000	0,750000	1,000000	0,398721	-0,695500	-2,281718	$[a, c]$	0,250000
2	0,500000	0,625000	0,750000	0,398721	-0,084879	-0,695500	$[a, c]$	0,125000
3	0,500000	0,562500	0,625000	0,398721	0,173023	-0,084879	$[c, b]$	0,062500
4	0,562500	0,593750	0,625000	0,173023	0,048071	-0,084879	$[a, c]$	0,031250
5	0,593750	0,609375	0,625000	0,048071	-0,017408	-0,084879	$[c, b]$	0,015625
6	0,593750	0,601563	0,609375	0,048071	0,015581	-0,017408	$[a, c]$	0,007813
7	0,601563	0,605469	0,609375	0,015581	-0,000851	-0,017408	$[c, b]$	0,003906
8	0,601563	0,603516	0,605469	0,015581	0.007380	-0,000851	$[c, b]$	0,001953
9	0,603516	0,604492	0,605469	0,007380	0,003268	-0,000851	$[c, b]$	0,000977
10	0,604492	0,604980	0,605469	0,003268	0,001210	-0,000851	$[c, b]$	0,000488

11	0,604980	0,605225	0,605469	0,001210	0,000179	-0,000851	$[c, b]$	0,000244
12	0,604980	0,605347	0,605469	0,000179	-0,000336	-0,000851	$[a, c]$	0,000122
13	0,605225	0,605286	0,605347	0,000179	-0,000078	-0,000336	$[a, c]$	0,000061
14	0,605225	0,605255	0,605286	0,000179	0,000051	-0,000078	$[c, b]$	0,000031
15	0,605225	0,605270	0,605286	0,000051	-0,000014	-0,000078	$[a, c]$	0,000015
16	0,605225	0,605263	0,605270	0,000051	0,000018	-0,000014	$[c, b]$	0,000008

Jadi hampiran akarnya adalah $x = 0{,}605263$

Jumlah lelaran yang dibutuhkan adalah

$$R > \frac{\ln(|1-0|) - \ln(0{,}00001)}{\ln 2}$$

$$R > 16{,}60964$$

Jadi dibutuhkan minimal 17 kali lelaran ($\mathrm{r} = 0$ sampai dengan $\mathrm{r} = 16$). Sesuai dengan jumlah lelaran pada tabel agar galat akar hampiran kurang dari ε.

2) Metode Regula Falsi

Meskipun metode bagi dua selalu berhasil menemukan akar, tetapi kecepatan konvergensinya sangat lambat. Kecepatan konvergensinya dapat ditingkatkan bila nilai $f(a)$ dan $f(b)$ juga turut diperhitungkan. Logikanya, bila $f(a)$ lebih dekat ke nol daripada $f(b)$ tentu akar lebih dekat ke $x = a$ daripada ke $x = b$. Metode yang memanfaatkan nilai $f(a)$ dan $f(b)$ ini adalah **metode regula-falsi** (bahasa Latin) atau **metode posisi palsu.** *(false position method).* Dengan metode regula-falsi, dibuat garis lurus yang menghubungkan titik $(a, f(a))$ dan $(b, f(b))$. Perpotongan garis tersebut dengan sumbu-x merupakan taksiran akar yang diperbaiki. Garis lurus tadi seolah-olah berlaku menggantikan kurva $f(x)$ dan memberikan posisi palsu dari akar.

Perhatikan Gambar 3.7

Gradien garis AB = gradien garis BC

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

Yang dapat disederhakan menjadi

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

Algoritma regula-falsi hampir sama dengan algoritma bagidua kecuali pada perhitungan nilai c.

```
procedure regula_falsi(a, b: real);
{ Mencari akar f(x)=0 di dalam selang [a,b] dengan metode regulafalsi
K.Awal : a dan b adalah ujung-ujung selang sehingga f(a)*f(b) < 0,
harga a dan b sudah terdefenisi
K.Akhir : Hampiran akar tercetak di layar }
const
epsilon1 = 0.00001;   {batas lebar selang akhir lelaran}
epsilon2 = 0.000001;     {bilangan yang sangat kecil, bisa diganti}
begin
      repeat
      c:=b-(f(b)*(b-a)/(f(b)-f(a)));
      if abs(f(c))< epsilon2 then {f(c) = 0, c adalah akar}
            begin
              a:=c;
              b:=c;
            end
      else
        if f(a)*f(c) < 0 then
            b:=c; {selang baru [a,b]=[a,c]}
```

```
      else
        a:=c; {selang baru [a,b]=[c,b]}
      until ABS(a-b)< epsilon1;
      {c adalah hampiran akar }
      writeln('Hampiran akar : ', c:10:6);
end.
```

Secara umum, metode regula-falsi lebih cepat konvergensinya dibandingkan dengan metode bagidua. Namun, pada beberapa kasus kecepatan konvergensinya justru lebih lambat. Bila kita memakai Program 3.4 untuk menghitung akar $f(x) = e^x - 5x^2$ di dalam selang $[0,1]$ *dan* $\varepsilon = 0.00001$, maka tabel lelerannya yang menghasilkan adalah sebagai berikut:

r	a	c	b	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebarnya
0	0.000000	0.304718	1.000000	1.000000	0.891976	-2.281718	$[c,b]$	0.695282
1	0.304718	0.500129	1.000000	0.891976	0.398287	-2.281718	$[c,b]$	0.499871
2	0.500129	0.574417	1.000000	0.398287	0.126319	-2.281718	$[c,b]$	0.425583
3	0.574417	0.596742	1.000000	0.126319	0.035686	-2.281718	$[c,b]$	0.403258
4	0.596742	0.602952	1.000000	0.035686	0.009750	-2.281718	$[c,b]$	0.397048
5	0.602952	0.604641	1.000000	0.009750	0.002639	-2.281718	$[c,b]$	0.395359
6	0.604641	0.605098	1.000000	0.002639	0.000713	-2.281718	$[c,b]$	0.394902
7	0.605098	0.605222	1.000000	0.000713	0.000192	-2.281718	$[c,b]$	0.394778
8	0.605222	0.605255	1.000000	0.000192	0.000052	-2.281718	$[c,b]$	0.394745
9	0.605255	0.605264	1.000000	0.000052	0.000014	-2.281718	$[c,b]$	0.394736
10	0.605264	0.304718	1.000000	0.000014	0.000004	-2.281718	$[c,b]$	0.394734

11	0.605266	0.605267	1.000000	0.000004	0.000001	-2.281718	$[c, b]$	0.394733
12	0.605267	0.605267	1.000000	0.000001	0.000000	-2.281718	$[c, b]$	0.394733
13	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
14	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
15	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
16	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
17	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
18	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
19	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
20	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	$[c, b]$	0.394733
21	0.605267	0.605267	1.000000	0.000000	-0.000000	-2.281718	$[a, c]$	0.000000

Hampiran akar $x = 0.605267$

Jumlah lelaran tabel di atas = 22, lebih banyak daripada jumlah lelaran metode bagidua. Bila diperhatikan, dari lelaran 12 sampai lelaran 21, nilai a, b, c tidak pernah berubah, padahal $f(c)$ sudah sangat kecil (≈ 0). Kasus seperti ini akan terjadi bila kurva fungsinya cekung (konkaf) di dalam selang [a,b]. Akibatnya, garis potongnya selalu terletak di atas kurva (bila kurvanya cekung ke atas) atau selalu terletak dibawah kurva (bila kurvanya cekung ke bawah). Perhatikan gambar berikut.

Pada kondisi yang paling ekstrim, $|b - a_r|$ tidak pernah lebih kecil dari ε, sebab salah satu titik ujung selang, dalam hal ini b, selalu tetap untuk setiap lelaran $r = 0, 1, 2, ...$ Titik ujung selang yang tidak pernah berubah itu dinamakan **titik mandek** (*stagnant point*). Pada titik mandek

$$|b_r - a_r| = |b - a_r| \quad r = 0, 1, 2, ...$$

Yang dapat mengakibatkan program mengalami *looping* . Untuk mengatasi hal ini, kondisi berhenti pada algoritma regula-falsi harus kita tambah dengan memeriksa apakah nilai $f(c)$ sudah sangat kecil sehingga mendekati nol. Jadi, kondisi pada **repeat-until** menjadi

```
Until (ABS(a-b) < epsilon1) or (ABS f(c)) < epsilon2)
```

Bila perubahan ini diterapkan pada soal pencarian akar di atas dengan `epsilon2 = 0.000001`, lelarannya akan berhenti pada $r = 12$ dengan akar $x = 0.605267$

Perbaikan Metode Regula-Falsi

Untuk mengatasi kemungkinan kasus titik mandek, metode regula-falsi kemudian diperbaiki (*modified false position method).* Caranya, pada akhir lelaran r = 0, kita sudah memperoleh selang baru akan dipakai pada lelaran r = 1. Berdasarkan selang baru tersebut, tentukan titik ujung selang yang tidak berubah (jumlah perulangan > 1) yang kemudian menjadi titik mandek. Nilai *f* pada titik mandek itu diganti menjadi setengah kalinya, yang akan dipakai pada lelaran r = 1.

Misalnya fungsi $f(x)$ cekung ke atas di dalam selang [a, b] seperti yang ditunjukkan pada gambar 3.9

Setelah menghitung nilai c_0 pada lelaran $r = 0$, ujung selang b untuk lelaran $r = 1$ tidak berubah. Titik b menjadi titik mandek. Karena itu, untuk lelaran $r = 1$, nilai $f(b)$ yang dipakai adalah $f(b)/2$. Begitu juga untuk lelaran $r = 2$, nilai $f(b)$ yang dipakai adalah setengah dari nilai $f(b)$ sebelumnya. Pada akgir lelaran $r = 2$, c_2 sudah terletak di bawah kurva $y = f(x)$. Selang yang dipakai selanjutnya adalah $[c_1, c_2]$. Dengan cara ini kita dapat menghilangkan titik mandek yang berkepanjangan. Program diatas kita modifikasi menjadi seperti berikut

```
Procedure perbaikan_regulasi_falsi (a, b real);
{mencari akar f(x)=0 di dalam selang [a, b] dengan metode regula-falsi yang diperbaiki
      K.Awal:a dan b adalah ujung-ujung selang sehingga f(a)*f(b)<0;
      K.Akhir:akar persamaan tercetak di layar
}
Const
      Epsilon1=0.00001 {batas lebar selang akhir lelaran}
      Epsilon2=0.000001 {batas galat nilai fungsi di hampiran akar}
Var
      FA,FB,simpan:real;
      Mandek_kiri,mandek_kanan:integer; {jumlah perulangan titik ujung selang}
```

```
Begin
      FA:=f(a); FB:=f(b);
      Mandek_kiri:=1; mande_kanan:=1;
      Repeat
        C:=b-(FB*(b-a)/(FB-FA);

      if abs(f(c))< epsilon2 then {f(c) = 0, c adalah akar}
            begin
              a:=c;
              b:=c;
            end
      else
        if f(a)*f(c) < 0 then
            begin
            b:=c; {selang baru [a,b]=[a,c]}
            FB:=f(c);
            Mandek_kiri:=mandek_kiri+1;
            Mandek_kanan:=0;
            If mandek kiri>1 then
               FA:=FA/2;          {a menjadi titik mandek}
            End;
      Else
      Begin
        a:=c;                     {selang baru [a,b]=[c,b]}
        FA:=f(c);
        mandek_kanan:=mandek_kanan=1;
        mandek_kiri:=0;
        if mandek_kanan>1 then
```

Tabel lelaran dari program diatas untuk menghitung akar $f(x) = e^x - 5x^2$ di dalam selang [0,1] dengan $\varepsilon = 0.00001$ *dan* $\delta = 0.000001$ adalah sebagai berikut:

r	a	c	b	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebarnya
0	0.000000	0.304718	1.000000	1.000000	0.891976	-2.281718 ↓ (*/ 2)	$[c, b]$	0.695282
1	0.304718	0.609797	1.000000	0.891976	-0.019205	-1.140859	$[a, c]$	0.305079
2	0.304718	0.603367	0.609797	0.891976	0.008005	-0.019205	$[c, b]$	0.006430
3	0.603367	0.605259	0.609797	0.008005	0.000035	-0.019205 ↓ (*/ 2)	$[c, b]$	0.004538
4	0.605259	0.605275	0.609797	0.000035	-0.000035	-0.009602	$[a, c]$	0.000017
5	0.605259	0.605267	0.605275	0.000035	0.000000	-0.000035	$[c, b]$	0.000008

Hampiran akar $x = 0.605267$

Terlihat bahwa jumlah lelarannya berkurang menjadi sepertiga semula. Harus dicatat bahwa metode regula-falsi yang diperbaiki tetap berlaku untuk fungsi yang tidak cekung sekalipun. Jadi, jika anda memprogram dengan metode regula-falsi, pakailah Program 3.4 ini untuk semua kemungkinan kasus fungsi.

C. Metode Terbuka

Metode ini tidak memerlukan selang yang mengurung akar. Yang diperlukan hanyalah sebuah tebakan awal akar atau dua buah tebakan yang tidak perlu mengurung akar. Oleh karena itulah metodenya dinamakan metode terbuka.

Yang termasuk dalam metode terbuka adalah :

1. Metode lelaran titik tetap (*fixed-point interation*)
2. Metode Newton-Raphson
3. Metode *secant*

1) Metode Lelaran Titik Tetap

Metode ini disebut juga metode lelaran sederhana, metode langsung atau metode sulih beruntun. Pembentukan prosedur lelarannya adalah sebagai berikut ;

Susunlah persamaan $f(x) = 0$ menjadi bentuk menjadi bentuk $x = g(x)$. Kemudian bentuk menjadi prosedur lelaran

$$x_{r+1} = g(x_r); r = 1,2,3, \ldots$$

dan terka sebuah nilai awal x_0, lalu hitung nilai $x_1, x_2, x_{3,\ldots}$ yang mudah-mudahan konvergen ke akar sejati *s* sedemikian sehingga

$$f(s) = 0 \text{ dan } s = g(s).$$

Kondisi berhenniti lelaran dinyatakan bila

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

atau bila menggunakan gelat relatif hampiran

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

dengan ε dan δ telah ditetapkan sebelumnya. Program lelaran titik-tetap ditunjukkan oleh program di bawah.

Program Metode lelaran titik tetap

```
procedure lelaran_titik_tetap(x:real);

{ Mencari akar persamaan f(x) = 0 dengan metode lelaran
titik - tetap

K.Awal : x0 dan x1 adalah tebakan awal akar, terdefenisi
nilainya

K.Akhir: akar persamaan tercetak di layar  }

const

epsilon = 0.000001;

var

 x_sebelumnya: real;

function g(x:real):real;

{ mengembalikan nilai g(x). Definisi g(x) bergantung pada
```

```
persoalan

begin

repeat

  x_sebelumnya:=x;

x:=g(x);

until (ABS(x-x_sebelumnya) < epsilon);

 { x adalah hampiran akar persamaan }

write('Hampiran akar x = ', x:10:6);

end;
```

Program di atas hanya menangani lelaran yang konvergen. Program harus dimodifikasi menjadi seperti program yang di bawah untuk menangani lelaran yang divergen. Salah satu cara penanganannya adalah dengan membatasi jumlah maksimum lelaran (*Nmaks*). Jika jumlah lelaran lebih besar dari *Nmaks* , maka diasumsikan lelarannya divergen.

Program Metode lelaran titik-tetap (dengan penanganan kasus divergen)

```
procedure lelaran_titik_tetap(x:real);

{ Mencari akar persamaan f(x) = 0 dengan metode lelaran titik -
tetap

K.Awal : x0 dan x1 adalah tebakan awal akar, terdefenisi
nilainya

K.Akhir: akar persamaan tercetak di layar  }

const

epsilon = 0.000001;

Nmaks = 30;

var

 x_sebelumnya: real; {hampiran nilai akar pada lelaran
sebelumnya}

i : integer; { pencacah nilai jumlah }
```

```
function g(x:real):real;

{ mengembalikan nilai g(x). Definisi g(x) bergantung pada
persoalan

begin i:=0;

repeat

  x_sebelumnya:=x;

x:=g(x);

i:=i+1;

until (ABS(x-x_sebelumnya) < epsilon);

 { x adalah hampiran akar persamaan }

If i > Nmaks then

Write (Divergen!')

else

write('Hampiran akar x = ', x:10:6);

end;
```

Contoh 3.2:

Carilah akar persamaan $f(x) = x^2 - 2x - 3 = 0$ dengan metode lelaran titik-tetap. Gunakan $\varepsilon = 0.000001$.

Penyelesaian :

(a) $x^2 - 2x - 3 = 0$

$$x^2 = 2x + 3$$

$$x = \sqrt{2x + 3}$$

Dalam hal ini, $g(x) = \sqrt{2x + 3}$. Prosedur lelaranya adalah $x_{r+1} = \sqrt{2x_r + 3}$.

Ambil terkaan awal $x_0 = 4$

Tabel lelaranya :

r	x_r	$\lvert x_{r+1} - x_r \rvert$
0	4.000000	-

1	3.316625	0.683375
2	3.103748	0.212877
3	3.034385	0.069362
4	3.011440	0.022945
5	3.003811	0.007629
6	3.001270	0.002541
7	3.000423	0.000847
8	3.000141	0.000282
9	3.000047	0.000094
10	3.000016	0.000031
11	3.000005	0.000010
12	3.000002	0.000003
13	3.000001	0.000001
14	3.000000	0.000000

Hampiran akar x=3.000000 (konvergen monoton)

(b) $x^2 - 2x - 3 = 0$

$$x(x-2) = 3$$

$$x = 3/(x-2)$$

Dalam hal ini, $g(x) = 3/(x-2)$. Prosedur lelaranya adalah

$$x_{r+1} = \frac{3}{x_r - 2}$$

Ambil terkaan awal $x_0 = 4$.

Tabel lelaranya :

i	x_r	$\lvert x_{r+1} - x_r \rvert$
0	4.000000	-
1	1.500000	2.500000
2	-6.000000	7.500000
3	-0.375000	5.625000
4	-1.263158	0.888158
5	-0.919355	0.343803
6	-1.027624	0.108269
7	-0.990876	0.036748
8	-1.003051	0.012175
9	-0.998984	0.004066
10	-1.000339	0.001355
11	-0.999887	0.000452
12	-1.000038	0.000151
13	-0.999987	0.000050
14	-1.000004	0.000017
15	-0.999999	0.000006
16	-1.000000	0.000002
17	-1.000000	0.000001

Hampiran akar x = -1.000000 (konvergen berosilasi)

(c) $x^2 - 2x - 3 = 0$

$$x = (x^2 - 3)/2$$

Prosedur lelarannya adalah $x_{r+1} = (x_r{}^2 - 3)/2$. Ambil terkaan awal $x_0 = 4$.

Tabel lelaranya :

i	x_r	$\lvert x_{r+1} - x_r \rvert$
0	4.000000	- 0
1	6.500000	2.500000
2	19.625000	13.125000
3	191.070313	171.445312
4	18252.432159	18061.361847
...		

Ternyata lelarannya divergen!

Contoh 3.3 :

Apa yang terjadi dengan pemilihan beragam nilai x_0 pada pencarian akar persamaan $x^3 + 6x - 3 = 0$ dengan prosedur lelaran $x_{r+1} = \frac{-x^3+3}{6}$.

Cobakan dengan:

$x_0 = 0.5$

$x_0 = 1.5$

$x_0 = 2.2$

$x_0 = 2.7$

Penyelesaian

Tabel lelarannya sebagai berikut;

r	x_r	r	x_r	r	x_r	r	x_r
0	0.5	0	1.5	0	2.2	0	2.7
1	0.4791667	1	-0.0625	1	-1.2744667	1	-2.7805
2	0.4816638	2	0.5000407	2	0.8451745	2	4.0827578
3	0.4813757	3	0.4791616	3	0.3993792	3	-10.842521
...		4	0.4816644	4	0.4893829	4	212.9416
7	0.4814056	...		...		5	-16909274.5
8	0.4814056	9	0.4814056	9	0.4814054		
		10	0.4814056	10	0.4814056		
				11	0.4814056		

konvergen | divergen

Terlihat dengan pengambilan x_0 yang cukup dekat ke akar sejati, proses akar konvergen tetapi jika kita mengambil x_0 terlalu jauh dari akar sejati, maka akan divergen.

Kriteria Konvergensi Metode Lelaran Titik-Tetap

Diberikan prosedur lelaran

$$x_{r+1} = g(x_r)$$

Misalkan x = s adalah solusi *f(x)=0* sehingga *f(s)=0* dan $s = g(s)$. Selisih antara x_{r+1} dan s adalah

$$x_{r+1} - s = g(x_r) - s$$

$$= \frac{g(x_r)-s}{(x_r-s)}(x_r - s)$$

Terapkan teorema nilai rata-rata sehingga

$x_{r+1} - s = g'(t)(x_r - s)$

yang mana $x_{r+1} < t < s$. Misalkan galat pada lelaran ke-r dan lelaran ke-(r+1) adalah

$\varepsilon_r = x_r - s$ dan $\varepsilon_{r+1} = x_{r+1} - s$

dan dapat kita tulis menjadi

$\varepsilon_{r+1} = g'(t)\varepsilon_r$

atau dalam tanda mutlak

$|\varepsilon_{r+1}| = |g'(t)||\varepsilon_r| \leq K|\varepsilon_r|$

dimana $g'(x) \leq K < 1$.

Teorema 3.2

Misalkan $g(x)$dan $g'(x)$menerus di dalam selang [a,b] =[s-h,s+h] yang mengandung titik tetap s dan nilai awal x_0 dipilih dalam selang tersebut. Jika $|g'(x)| < 1$ untuk semua $x \in [a,b]$ maka lelaran $x_{r+1} = g(x_r)$ akan konvergen ke s. Pada kasus ini s disebut juga titik *atraaktif*. Jika Jika $|g'(x)| > 1$ untuk semua $x \in [a,b]$ maka lelaran $x_{r+1} = g(x_r)$ akan divergan dari s.

Teorema 3.2 dapat kita sarikan sebagai berikut:

Di dalam selang I = [s-h, s+h], dengan s titik tetap,

1. jika $0 < g'(x) < 1$ untuk setiap $x \in I$, maka lelaran konvergen monoton;
2. jika -1< g'(x) < 0 untuk setiap x ∈ I, maka lelaran 2. jika -1< g'(x) < 0 untuk setiap x∈I, maka lelaran konvergen bersosilasi;
3. jika g'(x) > 1 untuk setiap x ∈ I, maka lelaran divergen monoton;
4. jika g'(x) < -1 untuk setiap x ∈ I, maka lelaran divergrn berosilasi.

2) Metode Newton-Rophson

Metode ini paling sering dipakai dan disukai karena konvergensinya paling cepat diantara metode yang lainya.

Ada dua pendekatan dalam menurunkan rumus metode Newton-Rophson, yaitu:

(a) Penurunan rumus Newton-Rophson secara geometri

Dari gambar grafik 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 Newton-Raphson adalah

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

(b) Penurunan rumus Newton-Rophson dengan bantuan deret Taylor

Uraikan $f(x_{r+1})$ disekitar x_r ke dalam deret Taylor:

$$f(x_{r+1}) = f(x_r) + (x_{r+1} - x_r)c + \frac{(x_{r+1}-x_r)^2}{2} f''(t), x_r < t < x_{r+1}$$

yang bila dipotong sampai suku orde-2 saja menjadi

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

Karena kita mencari akar, maka $f(x_{r+1}) = 0$, sehingga

$$0 = f(x_r) + (x_{r+1} - x_r)f'(x_r)$$

atau

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

yang merupakan rumus metode Newton-Rophson.

Kondisi berhenti lelaran Newton-Rophson adalah bila

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

atau bila menggunakan gelat relative hampiran

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

dengan ε dan δ adalah toleransi galat yang diinginkan.

Catatan :

1. Jika terjadi $f'(x_r) = 0$, ulangi kembali perhitungan lelaran dengan x_0 yang lain.
2. Jika persamaan $f(x) = 0$ memiliki lebih dari satu akar, pemilihan x_0 yang berbeda – beda dapat menemukan akar yang lain.
3. Dapat pulaterjadilelaran konvergen ke akar yang berbeda dari yang diharapkan (seperti halnya pada metode lelaran titik-tetap).

Program Metode Newton – Raphson

```
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;
```

Kriteria Konvergensi metode Newton-Raphson

Bentuk umum prosedur lelaran metode terbuka,

$$x_{r+1} = g(x_r)$$

Karena metode Newton-Raphson termasukmetode terbuka, maka dalam hal ini,

$$g(x) = x - \frac{f(x)}{f'(x)}$$

Dengan mengingat syarat perlu agar lelaran konvergen adalah $|g'(x)| < 1$, maka

$$g'(x) = 1 - \frac{[f'(x)f'(x) - f(x)f''(x)]}{[f'(x)]^2}$$

$$= \frac{f(x)f''(x)}{[f'(x)]^2}$$

Karena itu, metode Newton-Raphson akan konvergen bila

$\left|\frac{f(x)f''(x)}{[f'(x)]^2}\right| < 1$, dengan syarat $f'(x) \neq 0$.

3) Orde Konvergensi Metode Terbuka

Prosedur lelaran pada setiap metode terbuka dapat ditulis dalam bentuk $x_{r+1} = g(x_r)$

misalnya pada metode Newton-Raphson $g(x_r) = x_r - \frac{f(x_r)}{f'(x_r)}$. Misalkan x_r adalah hampiran tetap akar sejati s sehingga $s = g(s)$. Maka, berdasarkan konsep galat $s = x_r + \varepsilon_r$ dengan ε_r adalah galat dari x_r. Uraikan $g(s)$ disekitar x_r:

$$g(s) = g(x_r) + g'(x_r)(s - x_r) + {}^1/_2\, g(x_r)(s - x_r)^2 + \cdots.$$

$$= g(x_r) + g'(x_r)\varepsilon_r + {}^1/_2\, g(x_r){\varepsilon_r}^2 + \cdots.$$

Kemudian kurangi dengan $x_{r+1} = g(x_r)$ sehingga diperoleh:

$$g(s) - x_{r+1} = g'(x_r)\varepsilon_r + {}^1/_2\, g(x_r){\varepsilon_r}^2 + \cdots$$

Karena $s = g(s)$, maka

$$s - x_{r+1} = g'(x_r)\varepsilon_r + {}^1/_2\, g(x_r){\varepsilon_r}^2 + \cdots$$

Misalkan $s - x_{r+1} = \varepsilon_{r+1}$, sehingga

$$\varepsilon_{r+1} = g'(x_r)\varepsilon_r + {}^1/_2\, g(x_r){\varepsilon_r}^2 + \cdots$$

Bilangan pangkat dari $\varepsilon_r$menunjukkan orde(atau laju) konvergensi prosedur lelaran:

(a) $\varepsilon_{r+1} \approx g'(t)\varepsilon_r\,, x_r < t < x_{r+1}$: Prosedur lelaran orde satu

(b) $\varepsilon_{r+1} \approx {}^1/_2\, g(x_r){\varepsilon_r}^2, x_r < t < x_{r+1}$: Prosedur lelaran orde dua

Metode Newton-Rephson termasuk dalam metode terbuka orde dua.

Orde konvergensi metode Newton-Raphson

Pada metode Newton-Raphson, $g(x_r) = x_r - \frac{f(x_r)}{f'(x_r)}$. Turunan pertama dari $g(x_r)$ adalah

$$g(x_r) = \frac{f(x_r)f''(x_r)}{[f'(x_r)]^2}$$

Jika x_r adalah akar persamaan $f(x) = 0$, maka $f(x_r) = 0$, sehingga

$$g'(x_r) = 0$$

Ini berarti metode Newton-Raphson paling sedikit berorde dua. Turunan kedua dari $g(x_r)$ adalah

$$g''(x_r) = f''(x_r)/f'(x_r)$$

Substutusikan persamaan diatas ke $\varepsilon_{r+1} \approx {}^1/_2\, g(x_r)\varepsilon_r{}^2, x_r < t < x_{r+1}$ diperoleh:

$$\varepsilon_{r+1} = \frac{f''(x_r)\varepsilon_r{}^2}{2f'(x_r)}$$

Persamaan ini mempunyai tiga arti, yaitu :

1. Galat lelaran sekarang sebanding dengan kuadrat galat lelaran sebelumnya. Jika galat lelaran sekarang misalnya0.001, maka pada lelaran berikutnya galatnya sebanding dengan 0.000001. Hal inilah yang menyebabkan metode Newton-Raphson sangat cepat menurunkan akar(jika lelaranya konvergen).
2. Jumlah angka bena akan berlipat dua pada tiap lelaran. Ini merupakan akibat dari omor satu diatas.
3. Orde konvergensi metode Newton-Raphson adalah kuadratik sehingga dinamakan juga metode kuadratik.

Cara lain untuk menentukan orde konvergensi metode Newton Raphson adalah dengan meneruskan penurunan rumus Newton-Raphson dari deret Taylornya. Perhatikan persamaan berikut:

$$f(x_{r+1}) = f(x_r) + (x_{r+1} - x_r)c + \frac{(x_{r+1} - x_r)^2}{2} f''(t), x_r < t < x_{r+1}$$

Bila $x_{r+1} = s$ sehingga $f(x_{x+1}) = f(s) = 0$, dalam hal ini s adalah akar sejati, maka didapat:

$$0 = f(x_r) + (s - x_r)f'(x_r) + \frac{(s - x_r)^2}{2} f''(t)$$

Kurangi dengan $0 = f(x_r) + (x_{r+1} - x_r)f'(x_r)$, didapat:

$$0 = (s - x_{r+1})f'(x_r) + \frac{(s - x_r)^2 f''(t)}{2}$$

Misalkan $s - x_{r+1} = \varepsilon_{r+1}$ dan $s - x_r = \varepsilon_r$, maka persamaan diatas dapat ditulis menjadi

$$\varepsilon_{r+1} f'(x_r) + \frac{\varepsilon_r{}^2 f''(t)}{2} = 0$$

atau

$$\varepsilon_{r+1} = \frac{f''(t)\varepsilon_r{}^2}{2f'(x_r)}$$

Pada proses pencarian akar pada metode Newton-Raphson, muncul kesulitan jika $|f'(x)|$ terlalu dekat ke nol, dan kita harus menggunakan bilangan berketelitian ganda untuk memperoleh $f(x)$ dan $f'(x)$ cukup teliti. Persamaan nirlanjar $f(x) = 0$ yang mempunyai kasus seperti ini dinamakan kondisi buruk.

4) Metode Secant

Prosedur lelaran metode Newton-Raphson memerlukan perhitungan turunan fungsi, f '(x). Sayangnya, tidak semua fungsi mudah dicari turunannya,terutama fungsi yang bentuknya rumit. Turunan fungsi dapat dihilangkan dengan cara menggantinya dengan bentuk lain yang ekivalen. Modifikasi metode Newton-Raphson ini dinamakan Metode Secant.

Berdasarkan grafik diatas dapat kita hitung gradien :

$$f'(x_r) = \frac{\Delta y}{\Delta x} = \frac{AC}{BC} = \frac{f(x_r) - f(x_{r-1})}{x_r - x_{r-1}}$$

Substitusikan persamaan tersebut ke

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

Sehingga diperoleh

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

yang merupakan prosedur lelaran metode secant.Dalam hal ini diperlukan dua buah tebakan awal akar, yaitu x_0 dan x_1. Kondisi berhenti lelaran adalah bila

$$|x_{r+1} - x_r| < \varepsilon \text{ (galat mutlak) atau } \left|\frac{x_{r+1} - x_r}{x_{r+1}}\right| < \delta \text{ (galat hampiran)}$$

dengan ε dan δ adalah toleransi galat.

Metode secant mirip dengan metode regular-falsi tetapi prinsip keduanya berbeda. Perbedaan tersebut dapat dilihat dalam tabel berikut:

Program Metode Secant

```
procedure Secant(x0, x1:real);

{ Mencari akar persamaan f(x) = 0 dengan metode secant

 K.Awal : x0 dan x1 adalah tebakan awal akar, terdefenisi
nilainya

K.Akhir: akar persamaan tercetak di layar

}

const

   epsilon = 0.000001;      { toleransi galat akar hampiran
}

var

   x_sebelumnya: real;

 function f(x:real):real;

  { mengembalikan nilai f(x). Definisi f(x) bergantung
pada persoal



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);

      { x adalah hampiran akar persamaan }

 write('Hampiran akar x = ', x:10:6);
```

```
end;
```

Contoh:

Hitunglah akar $f(x) = e^x - 5x^2$ dengan metode secant. Gunakan ε = 0.00001. Tebakan awal akar $x_0 = 0.5$ dan $x_1 = 1$

Penyelesaian:

Tabel lelarannya:

i	x_r	$\lvert x_{r+1} - x_r \rvert$
0	0.500000	-
1	1.000000	0.500000
3	-0.797042	1.797042
4	10.235035	11.032077
5	-0.795942	11.030977
6	-0.794846	0.001096
7	-0.472759	0.322087
8	-0.400829	0.071930
9	-0.374194	0.026635
10	-0.371501	0.002692
11	-0.371418	0.000083
12	-0.371418	0.000000

Akar $x = -0.371418$

Ternyata lelaranya mengarah ke akar yang lain, yaitu $x = -0.371418$.

D. Akar Ganda

Akar ganda berpadanan dengan suatu titik dimana fungsi menyinggung sumbu x. Misalnya, akar ganda-dua dihasilkan dari

$f(x) = (x-3)(x-1)(x-1)$ .....(*)

atau dengan mengalikan faktor-faktornya,

$f(x) = x^3 - 5x^2 + 7x - 3$

Persamaan tersebut mempunyai akar kembar karena satu nilai x menyebabkan dua faktor dalam Persamaan (*) sama dengan nol. Secara grafis, ini berpadanan

terhadap kurva yang menyentuh sumbu **x** secara bersinggungan pada akar kembar tersebut.

Akar ganda-tiga (triple root) berpadanan dengan kasus dimana satu nilai **x** membuat tiga faktor dalam suatu persamaan sama dengan nol, seperti dalam

$$f(x) = (x-3)(x-1)(x-1)(x-1)$$

atau,dengan mengalikan fakror-faktornya,

$$f(x) = x^4 - 6x^3 + 12x^2 - 10x - 3$$

Akar ganda menimbulkan sejumlah kesukaran untuk banyak metode numerik :

1. Kenyataan bahwa fungsi tidak berubah tanda pada akar ganda genap menghalangi penggunaan metode-metode tertutup. Metode terbuka, seperti metode Newton-Raphson, sebenarnya dapat diterapkan di sini. Tetapi, bila digunakan metode Newton Raphson untuk mencari akar ganda, kecepatan konvergensinya berjalan secara linear, tidak lagi kuadratis sebagaimana aslinya.
2. Permasalahan lain yang mungkin berkaitan dengan fakta bahwa tidak hanya $f(x)$ tetapi juga $f'(x)$ menuju ke nol pada akar. Ini menimbulkan masalah untuk metode Newton-raphson maupun metode secant (talibusur), yang dua-duanya mengandung turunan (atau taksirannya) pada penyebut rumus mereka masing-masing. Ini dapat menghasilkan pembagian oleh nol pada waktu penyelesaian konvergen sangat dekat ke akar. Pembagian dengan nol ini dapat dihindari dengan melihat fakta bahwa $f(x)$ lebih dulu nol sebelum $f'(x)$. Jadi

   ```
   if f(x) = 0 then hentikan lelaran
   ```

3. Ralston dan Rabinowitz (1978) telah menunjukkan bahwa perubahan sedikit dalam perumusan mengembalikannya ke kekonvergenan kuadrat, seperti dalam

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)} \dots\dots (**)$$

Dimana m adalah multiplisitas akar (yaitu, $\mathrm{m} = 2$ untuk akar kembar, $\mathrm{m} = 3$ untuk akar ganda-tiga, dan seterusnya). Tentu saja, ini mungkin merupakan alternative yang tidak memuaskan karena bergantung pada pengetahuan sebelumnya tentang multiplisitas akar.

Alternatif lain yang juga disarankan oleh Ralston dan Rabinowitz(1978) adalah mendefinisikan suatu fungsi baru $\mathrm{u}(\mathrm{x})$, yaitu rasio (hasil bagi) fungsi terhadap turunannya seperti dalam

$$u(x) = \frac{f(x)}{f'(x)} \dots\dots (***)$$

Dapat diperlihatkan bahwa fungsi ini mempunyai akar pada lokasi yang sama seperti fungsi semula. Oleh karena itu, persamaan di atas dapat disubtitusikan ke dalam persamaan (**) dengan maksud mengembangkan suatu bentuk alternatif dari metode Newton-Raphson :

$$x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)} \dots\dots (****)$$

Persamaan (***) dapat didiferensialkan untuk memberikan

$$u'(x) = \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2} \dots\dots (*****)$$

Persamaan (***) dan (*****) dapat disubstitusikan ke dalam Persamaan (****) dan hasilnya disederhanakan untuk menghasilkan

$$x_{i+1} = x_i + \frac{f(x_i)f'(x_i)}{[f'(x_i]^2 - f(x_i)f''(x_i)}$$

Contoh:

Metode Newton-Raphson yang dimodifikasi untuk Akar Ganda

Pernyataan masalah : Gunakan baik metode Newton-Raphson yang baku maupun yang dimodifikasi untuk menghitung akar ganda dari $f(x) = \mathrm{x}^3 - 5\mathrm{x}^2 + 7\mathrm{x} - 3$, dengan terkaan awal $\mathrm{x}_0 = 0$.

Penyelesaian : $f(x) = \mathrm{x}^3 - 5\mathrm{x}^2 + 7\mathrm{x} - 3$

$$f'(x) = 3x^2 - 10x + 7$$

$$f''(x) = 6x - 10$$

Dengan metode Newton-Raphson yang dimodifikasi :

$$x_{i+1} = x_i - \frac{(x_i^3 - 5x_i^2 + 7x_i - 3)(3x_i^2 - 10x_i + 7)}{(3x_i^2 - 10x_i + 7)^2 - (6x_i - 10)(x_i^3 - 5x_i^2 + 7x_i - 3)}$$

Dengan metode Newton-Raphson yang baku :

$$x_{i+1} = x_i - \frac{(x_i^3 - 5x_i^2 + 7x_i - 3)}{(3x_i^2 - 10x_i + 7)}$$

Tabel lelarannya adalah:

Metode Newton – Raphson baku		Metode Newton – Raphson yang di modifikasi	
i	x_i	i	x_i
0	0.000000000	0	0.000000000
1	0.428571429	1	1.105263158
2	0.685714286	2	1.003081664
3	0.832865400	3	1.000002382
4	0.913328983		
5	0.955783293		
6	0.977655101		

Lelaran konvergen ke akar $\mathbf{x = 1}$. Terlihat dari tabel di atas bahwa metode newton yang dimodofikasi memiliki jumlah lelaran lebih sedikit.

E. Akar-Akar Polinom

Polinom adalah persamaan matematika berderajat-n, dengan kata lain pernyataan matematika yang melibatkan penjumlahan, perkalian, dan pemangkatan dalam satu atau lebih variabel dengan koefisien. Bentuk baku polinom derajat ≤ *n* adalah

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

dengan a_i adalah konstanta riil, $i = 0, 1, 2, \ldots, n$, dan $a_n \neq 0$. Polinom *p(x)* memiliki *n* buah akar, baik akar nyata maupun akar kompleks. Akar kompleks

muncul dalam pasangan konjugasi, $w = u + vi$ dan $w = u - vi$. Dengan $i = \sqrt{-1}$. Contohnya polinom $p(x) = 5 - 4x + x^2$ mempunyai akar $2 + i$ dan $2 - i$.

Semua metode pencarian akar dapat diterapkan pada polinom. Misalnya dengan metode Newton-Raphson :

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

Semakin tinggi derajat polinomnya tentu semakin banyak operasi perkalian yang diperlukan, yang berarti semakin besar rambatan galat pembulatannya. Karena itu, harus dicari suatu metode perhitungan polinom dengan sedikit operasi perkalian.

1. Metode Horner untuk Evaluasi Polinom

Metode Horner, atau disebut juga metode perkalian bersarang (*nested multiplication*) menyediakan cara perhitungan polinom dengan sedikit operasi perkalian. Dalam hal ini, polinom *p(x)* dinyatakan sebagai perkalian bersarang

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-1} + a_n x)\ldots)))$$

Contoh1:

Nyatakan $p(x) = 8 + 6x + 2x^2 + 5x^3$ dalam bentuk perkalian bersarang.

Penyelesaian:

$p(x) = 8 + 6x + 2x^2 + 5x^3$ (6 buah perkalian)

$= 8 + x(6 + x(2 + 5x))$ (hanya 3 buah perkalian)

Perhitungan *p(x)* untuk $x = 2$ adalah

$p(2) = 8 + 2(6 + 2(2 + 5.2)) = 68$

Metode perkalian bersarang untuk menghitung *p(t)* seringkali dinyatakan dalam bentuk table Horner berikut:

t	a_n	a_{n-1}	…	a_1	a_0
		tb_n	…	tb_2	tb_1
	$b_n=a_n$	$b_{n-1}=a_{n-1}+tb_n$		$b_1=a_1+tb_2$	$b_0=a_0+tb_1$

Polinom sisa

hasil evaluasi: $p(t) = b_0$

Jadi, untuk Contoh1 di atas,

2	5	2	6	8
		10	24	60
	5	12	30	68 = *p*(2)

Dan menghasilkan polinom sisa $5x^2 + 12x + 30$.

Program1: Menghitung $p(x)$ untuk $x = t$ dengan metode Horner

```
{  Dalam program utama telah didefinisikan:
   const n=...; {derajat polinom}
   var a, b, c: array[1..n]of real
}
function p(t:real):real;
{ menghitung p(t) dengan metode Horner }
var
      k: integer;
begin
      b[n]:=a[n];
      for k:=n-1 downto 0 do
            b[k]:=a[k] + b[k+1] * t;
      {end for}
      p:=b[0];
end;
```

2. Pencarian Akar-Akar Polinom

Proses perhitungan *p(x)* untuk *x* = *t* dengan metode Horner sering dinamakan *pembagian sintetis p(x) : x-t*, menghasilkan *q(x)* dan sisa b_0 .

$$\left[\frac{p(x)}{(x-t)} = q(x)\right] + \text{sisa } b_0$$

atau

$$p(x) = b_0 + (x-t)q(x)$$

yang dalam hal ini,

$$q(x) = b_n x^{n-1} + b_{n-1}x^{n-2} + \cdots + b_3 x^2 + b_2 x + b_1.$$

Untuk contoh1 di atas,

$$p(x) = 8 + 6x + 2x^2 + 5x^3 = 68 + (x-2)(5x^2 + 12x + 30)$$

Jika *t* adalah hampiran akar polinom *p(x)* maka

$$p(t) = b_0 + (t-t)q(t) = b_0 + 0 = b_0$$

(Perhatikan, jika *t* akar sejati, maka $b_0 = 0$)

Akar-akar lain dari *p(x)* dapat dicari dari polinom *q(x)* sebab setiap akar *q(x)* juga adalah akar *p(x)*. Proses reduksi polinom ini disebut *deflasi (deflation)*. Koefisien-koefisien *q(x)*, yaitu b_n, b_{n-1}, …, b_3, b_2, b_1 dapat ditemukan langsung dari tabel Horner.

$$bn = an$$

$$bn - 1 = an - 1 + tbn$$

...

$$b1 = a1 + tb2$$

Algoritmanya,

```
b[n]:=a[n];
for k:=n-1 downto 1 do
 b[k]:=a[k]+t*b[k+1]
{endfor}
```

Misalkan akar polinom dihitung dengan metode Newton-Raphson,

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

maka proses pencarian akar secara diflasi dapat dirumuskan dalam langkah 1 sampai 4 berikut ini.

Langkah 1;

Menghitung $p(x_r)$ dapat dilakukan dengan metode Horner.

Misalkan $t = x_r$ adalah hampiran akar polinom *p(x)*,

$$p(x) = b_0 + (x - x_r)q(x)$$

Perhitungan $p(x_r)$ menghasilkan

$$p(x_r) = b_0 + (x_r - x_r)q(x_r) = b_0$$

Nilai $p(x_r) = b_0$ ini dapat dihitung dengan `function p`

Langkah 2;

Menghitung $p'(x_r)$:

Misalkan t = x, adalah hampiran akar polinom p(x)

$$p(x) = b_0 + (x - x_r)q(x)$$

Turunan dari *p* adalah

$$p'(x) = 0 + 1.q(x) + (x - x_r)q'(x) = q(x) + (x - x_r)q'(x)$$

sehingga,

$$p'(x_r) = q(x_r) + (x_r - x_r)q'(x_r) = q(x_r)$$

Koefisien polinom *q(x)* dapat ditentukan dari langkah 1. Selanjutnya $q(x_r)$ dapat dihitung dengan `function q` berikut:

Program2: Menghitung $\mathbf{p'(t) = q(t)}$

```
  Function q(t:real):real;
{Menghitung p'(t)=q(t) dengan metode Horner}
Var
      k: integer;
begin
      c[n]:=b[n];
      for k:=n-1 downto 1 do
            c[k]:=b[k] + t*c[k-1]
      {end for}
      q:=c[1];
end;
```

Langkah 3;

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

Langkah 4;

Ulangi langkah 1, 2, dan 3 di atas $|x_{r+1} - x_r| < \varepsilon$.

Program3: Prosedur Newton_Raphson untuk menghitung akar polinom

```
Procedure Newton_Raphson_untuk_polinom(n:integer; x:real);
{ procedure Newton_Raphson untuk menghitung akar polinom
p(x) yang
  berderajat n dengan tebakan awal akar x
  K.Awal  : n adalah derajat polinom; x adalah tebakan awal
akar;
```

```
                Kedua nilai sudah terdefinisi
  K.akhir : Hampiran akar polinom tercetak di layar.
}
Const
  epsilon = 0.0000001;
var
  x_sebelumnya : real;

   function p(t:real):real;
   {menghitung  p(t) dengan metode Horner}
   var
      k: integer;
   begin
      b[n]:=a[n];
      for k;=n-1 downto 0 do
        b[k]:=a[k] + b[k+1] * t;
      {end for}
      p:=b[0];
   end {p};

   function q(t:real):real;
   {menghitung p'(t)=q(t) dengan metode Horner}
   var
      k: integer;
   begin
      c[n]:=b[n];
      for k;=n-1 downto 1 do
        c[k]:=b[k] + t *c[k+1];
      {end for}
      q:=c[1];
   end {q};

begin
   repeat
      x_sebelumnya:=x;
      x:=x - p(x)/q(x);
```

```
    until ABS(x - x_sebelumnya) < epsilon;

    {x adalah akar polinom}
    Writeln('Hampiran akar = ', x:10:6);
end;
```

Program3 hanya menemukan satu buah akar polinom. Untuk mencari seluruh akar nyata polinom, harus dilakukan proses deflasi. Setelah akar pertama x_1 diperoleh, polinom *p(x)* dapat ditulis sebagai

$$p(x) = (x - x_1)q(x) + b_0$$

yang dalam hal ini $q(x) = b_n x^{n-1} + b_{n-1}x^{n-2} + \cdots + b_3 x^2 + b_2 x + b_1$. Koefisien-koefisien $q(x)$, yaitu $bn, bn-1, \ldots, b3, b2, b1$ diperoleh di akhir Program1, yang telah tersimpan pada elemen larik (Nilai-nilai data dari kumpulan data yang bertipe tertentu dengan sebuah nama yang sama) $b[n-1], b[n-2], \ldots, b[2], b[1]$. Selanjutnya panggil Program3 untuk mencari akar polinom $q(x)$ yang berderajat *n*-1 dengan tebakan awalnya dapat digunakan x_1 (atau boleh bilangan lain). Setelah akar kedua x_2 diperoleh, polinom $p(x)$ dapat ditulis sebagai

$$q(x) = (x - x_2)r(x) + b_1$$

yang dalam hal ini

$$r(x) = b_{n-1}x^{n-2} + b_{n-2}x^{n-3} + \cdots + b_3 x^2 + b_2 x + b_1.$$

$r(x)$ adalah polinom derajat $n-2$ dengan koefisien b_{n-1}, b_{n-2}, …, b_3, b_2, b_1 diperoleh di akhir Program1 pada elemen larik $b[n-1], b[n-2], \ldots, b[2], b[1]$. Selanjutnya panggil kembali Progra3 untuk mencari akar polinom $r(x)$ yang berderajat $n-2$ dengan tebakan awalnya dapat digunakan x_2. Begitu seterusnya sampai polinom sisa yang ditemukan berderajat 0. Atau, dapat juga sampai polinom sisa berderajat dua.

Algoritma selengkapnya adalah:

```
write ('Tebakan awal untuk akar pertama: ');
readln(x);
repeat
   Newton_Raphson_untuk_polinom(n,x);

   {saling koefisien b[n], b[n-1], …, b[1] ke dalam
    a[n-1], a[n-2], …,a[0] untuk pencarian akar
    selanjutnya}
   for i:=n downto 1 do
    a[i-1]:=b[i];
   {end for}
   n:=n-1;   {derajat polinom sisa berkurang satu}
Until n:=0;
```

Contoh2:

Temukan seluruh akar nyata polinom

$p(x) = x^5 - 12x^4 - 293x^3 + 3444x^2 + 20884x - 240240$ dengan tebakan awal akar $x_0 = 11$.

Penyelesaian:

Panggil prosedur

```
Newton_Raphson_untuk_polinom(5,11);
```

untuk mencari akar polinom $p(x)$ berderajat 5 dengan tebakan awal akar $x_0 = 11$. Diperoleh akar pertama, yaitu $x_1 = 13{,}99990$

Deflasi → $p(x) = (x - x_1)q(x) + b_0$

yang dalam hal ini

$q(x) = x^4 + 1999895x^3 - 2650015x^2 - 2659927x + 17160.13$

Panggil prosedur

```
Newton_Raphson_untuk_polinom(4, 13.99990);
```

untuk mencari akar polinom $q(x)$ berderajat 4 dengan tebakan awal akar $x_0 = 13.99990$. Diperoleh akar kedua, yaitu $x_2 = 12{,}00016$

Deflasi → $q(x) = (x - x_2)r(x) + b_1$

yang dalam hal ini $r(x) = x^3 + 1400005x^2 - 9699867x - 1429992$

Panggil prosedur

```
Newton_Raphson_untuk_polinom(3, 12.00016);
```

Untuk mencari akar polinom $r(x)$ berderajat 3 dengan tebakan awal akar $x_0 = 12.00016$

Diperoleh akar $x_3 = 9.999949$

Deflasi → $r(x) = (x - x_3)s(x) + b_2$

yang dalam hal ini $s(x) = x^2 + 2396998x + 1429999$. Demikian seterusnya sampai ditemukan akar keempat dan akar kelima sebagai berikut :
$x_4 = -12.99991$, $x_5 = 11.00006$.

3. Lokasi Akar Polinom

Metode Newton_Raphson memerlukan tebakan awal akar. Misalkan akar-akar diberi indeks dan diurutkan menaik sedemikian sehingga

$$|x_1| \leq |x_2| \leq |x_3| \leq \cdots \leq |x_n|$$

Tebakan awal untuk akar terkecil x_1 menggunakan hampiran

$$a_0 + a_1 x \approx 0$$

$$x \approx -a_0/a_1$$

yang dapat dijadikan sebagai tebakan awal untuk menemukan x_1. Tebakan awal untuk akar terbesar x_n menggunakan hampiran

$$a_{n-1}x^{n-1} + a_n x^n \approx 0$$

$$x \approx -a_{n-1}/a_n$$

yang dapat dijadikan sebagai tebakan awal untuk menemukan x_n.

Contoh3:

Tentukan tebakan awal untuk mencari akar polinom $x^2 - 200x + 1 = 0$.

Penyelesaian:

Tebakan awal untuk akar terkecil adalah $x_0 = -\frac{1}{-200} = \frac{1}{200}$

Tebakan awal untuk akar terbesar adalah $x_0 = -\frac{-200}{1} = 200$

F. Sistem Persamaan Non Linear

Di dalam dunia nyata, umumnya model matematika muncul dalam bentuk sistem persamaan. Persamaan yang diselesaikan tidak hanya satu, tetapi dapat lebih dari sau, sehingga membentuk sebuah sistem yang disebut sistem persamaan non linear. Bentuk umum persamaan non linear dapat ditulis sebagai berikut:

$$f_1(x_1, x_2, \ldots, x_n) = 0$$
$$f_2(x_1, x_2, \ldots, x_n) = 0$$
$$\ldots$$
$$f_n(x_1, x_2, \ldots, x_n) = 0$$

Penyelesaian sistem ini adalah himpunan nilai *x* simultan, $x_1, x_2, \ldots, x_n$ yang memenuhi seluruh persamaan. Sistem persamaan dapat diselesaikan secara berlelar dengan metode lelaran (iterasi) titik-tetap atau dengan metode Newton-Raphson.

1. Metode Lelaran (Iterasi) Titik-Tetap

Prosedur lelarannya titik-tetap untuk sistem dengan dua persamaan non linear:

$$x_{r+1} = g_1(x_r, y_r)$$
$$y_{r+1} = g_2(x_r, y_r)$$
$$r = 0, 1, 2 \ldots$$

Metode lelaran titik-tetap seperti ini dinamakan metode **lelaran Jacobi**. Kondisi berhenti (konvergen) adalah

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

Kecepatan konvergensi lelaran titik-tetap ini dapat ditingkatkan. Nilai x_{r+1} yang baru dihitung langsung dipakai untuk menghitung y_{r+1}. Jadi,

$$x_{r+1} = g_1(x_r, y_r)$$
$$y_{r+1} = g_2(x_{r+1}, y_r)$$
$$r = 0, 1, 2 \ldots$$

Metode lelaran titik-tetap seperti ini dinamakan metode **lelaran Seidel**. Kondisi berhenti (konvergen) adalah

$$|x_{r+1} - x_r| < \varepsilon \text{ dan } |y_{r+1} - y_r| < \varepsilon$$

Untuk fungsi dengan tiga persamaan non linear, lelaran Seidelnya adalah

$$x_{r+1} = g_1(x_r, y_r, z_r)$$
$$y_{r+1} = g_2(x_{r+1}, y_r, z_r)$$
$$z_{r+1} = g_3(x_{r+1}, y_{r+1}, z_r)$$
$$r = 0, 1, 2 \ldots$$

Kondisi berhenti (konvergen) adalah

$$|x_{r+1} - x_r| < \varepsilon \text{ dan } |y_{r+1} - y_r| < \varepsilon \text{ dan } |z_{r+1} - z_r| < \varepsilon$$

Contoh

Masalah: Selesaikan sistem persamaan non linear berikut ini:

$$f_1(x, y) = x^2 + xy - 10 = 0$$
$$f_2(x, y) = y + 3xy^2 - 57 = 0$$

(Akar sejatinya adalah $x = 2$ dan $y = 3$)

Penyelesaian:

Persamaan diatas dapat dipecah untuk

$$x_{r+1} = \frac{10 - {x_r}^2}{y_r}$$

$$y_{r+1} = 57 - 3x_{r+1}{y_r}^2$$

Berikan tebakan awal $x_0 = 1.5$ dan $y_0 = 3.5$ dan $\varepsilon = 0.000001$

Tabel lelarannya

r	x	y	$\lvert x_{r+1} - x_r \rvert$	$\lvert y_{r+1} - y_r \rvert$
0	1.500000	3.50000	-	-
1	2.214286	-24.375000	0.714286	27.875000
2	-0.209105	429.713648	2,423391	454.088648
3	0.023170	-12778.041781	0.232275	13207.755429
…				

Ternyata lelarannya divergen!

Sekarang kita ubah prosedur lelarannya menjadi

$$x_{r+1} = \sqrt{10 - x_r y_r}$$

$$y_{r+1} = \sqrt{\frac{57 - y_r}{3x_{r+1}}}$$

Tebakan awal $x_0 = 1.5$ dan $y_0 = 3.5$ dan $\varepsilon = 0.000001$

Hasilnya,

r	x	y	$\lvert x_{r+1} - x_r \rvert$	$\lvert y_{r+1} - y_r \rvert$
0	1.500000	3.500000	-	-
1	2.179449	2.860506	0.679449	0.639494
2	1.940534	3.049551	0.238916	0.189045
3	2.020456	2.983405	0.079922	0.066146
4	1.993028	3.005704	0.0237428	0.022300
5	2.000279	2.998054	0.009357	0.007650
6	1.999905	3.000666	0.003200	0.002611
7	2.000033	2.999773	0.001094	0.000893
8	1.999905	3.000078	0.000374	0.000305

9	2.000033	2.999973	0.000128	0.000104
10	1.999989	3.000009	0.000044	0.000036
11	2.000004	2.999997	0.000015	0.000012
12	1.999999	3.000001	0.00005	0.000004
13	2.000000	3.000000	0.000002	0.000001
14	2.000000	3.000000	0.000001	0.00000

Akar $x = 2.000000$

$y = 3.000000$

2. Metode Newton-Raphson

Ingatlah kembali bahwa metode Newton-Raphson didasarkanpada pemakain turunan (kemiringan) suatu fungsi untuk menaksir perpotongannya dengan sumbu peubah bebasnya, yakni akar. Taksiran ini didasarkan dari deret Taylor.

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

dimana x_r adalah tebakan awal pada akarnya dan x_{r+1} adalah titk tempat garis singgung memotong sumbu *x*. Dan karena persoalan mencari akar, maka $f(x_{r+1}) = 0$, sehingga

$$0 = f(x_r) + (x_{r+1} - x_r)f'(x_r)$$

atau

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

yang merupakan bentuk persamaan tunggal dati metode Newton-Raphson. Untuk fungsi dengan dua peubah, deret Taylor orde pertama dapat dituliskan untuk masing-masing persamaan sebagai

$$u_{r+1} = u_r + (x_{r+1} - x_r)\frac{\partial u_r}{\partial x} + (y_{r+1} - y_r)\frac{\partial u_r}{\partial y}$$

dan

$$v_{r+1} = v_r + (x_{r+1} - x_r)\frac{\partial v_r}{\partial x} + (y_{r+1} - y_r)\frac{\partial v_r}{\partial y}$$

Sama halnya seperti untuk versi persamaan tunggal, taksiran akar berpadanan dengan titik-titik pada mana $u_{r+1} = 0$ dan $v_{r+1} = 0$, untuk memberikan

$$\frac{\partial u_r}{\partial_x}x_{r+1} + \frac{\partial u_r}{\partial_y}y_{r+1} = -u_r + x_r\frac{\partial u_r}{\partial_x} + y_r\frac{\partial u_r}{\partial_y}$$

$$\frac{\partial u_r}{\partial_x}x_{r+1} + \frac{\partial u_r}{\partial_y}y_{r+1} = -u_r + x_r\frac{\partial u_r}{\partial_x} + y_r\frac{\partial u_r}{\partial_y}$$

Dengan sedikit menerapkan manipulasi aljabar (misalnya aturan Cramer), kedua persamaan ini dapat dipecahkan menjadi

$$x_{r+1} = x_r - \frac{u_r \frac{\partial v_r}{\partial y} + v_r \frac{\partial u_r}{\partial y}}{\frac{\partial u_r}{\partial x}\frac{\partial v_r}{\partial y} - \frac{\partial u_r}{\partial y}\frac{\partial v_r}{\partial x}}$$

dan

$$y_{r+1} = y_r + \frac{u_r \frac{\partial v_r}{\partial x} - v_r \frac{\partial u_r}{\partial x}}{\frac{\partial u_r}{\partial x}\frac{\partial v_r}{\partial y} - \frac{\partial u_r}{\partial y}\frac{\partial v_r}{\partial x}}$$

(persamaan 5.21)

Penyebut dari masing-masing persamaan ini secara formal diacu sebagai **determinan Jacobi** dari persamaan tersebut.

Persamaan diatas adalah versi dua persamaan dari metode newton-Raphson. Seperti dalam contoh berikut, persamaan-persamaan tersebut dapat diterapkan secara iterative untuk secara simultan berakhir pada akar-akar dari dua persamaan simultan tersebut.

Contoh

Masalah: Gunakan metode newton-Raphson persamaan majemuk untuk menetukan akar-akar dari persamaan

$$f_1(x,y) = x^2 + xy - 10 = 0$$
$$f_2(x,y) = y + 3xy^2 - 57 = 0$$

Catat bahwa sepasang akar sejatinya adalah $x = 2$ dan $y = 3$. Awali komputasi dengan tebakan$x_0 = 1.5$ dan $y_0 = 3.5$.

Penyelesaian: Pertama kita hitung turunan-turunan parsial dan hitung nilainya pada tebakan-tebakan awal:

$$\frac{\partial u_0}{\partial x} = 2x + y = 2(1.5) + 3.5 = 6.5$$

$$\frac{\partial u_0}{\partial y} = x = 1.5$$

$$\frac{\partial v_0}{\partial x} = 3y^2 = 3(3.5)^2 = 36.75$$

$$\frac{\partial v_0}{\partial y} = 1 + 6xy = 1 + 6(1.5) = 32.5$$

Jadi determinan Jacobi untuk iterasi pertama adalah

$$6.5(32.5) - 1.5(36.75) = 156.125$$

Nilai-nilai fungsi dapat dihitung pada tebakan-tebakan awal sebagai

$$u_0 = (1.5)^2 + 1.5(3.5) - 10 = -2.5$$

$$v_0 = 3.5 + 3(1.5)(3.5)^2 - 57 = 1.625$$

Nilai-nilai ini dapat didistribusikan ke persamaan (5.21) untuk memberikan

$$x_0 = 1.5 - \frac{-2.5(32.5) - 1.625(1.5)}{156.125} = 2.03603$$

dan

$$y_0 = 3.5 + \frac{-2.5(36.75) - 1.625(6.5)}{156.125} = 2.84388$$

Jadi, hasil-hasilnya konvergen pada akar-akar sejati $x = 2$ dan $y = 3$. Komputasi dapat diulang sampai diperoleh kecermatan yang dapat diterima.

Sama halnya seperti dengan iterasi satu-titik, pendekatan Newton-Raphson seringkali akan divergen jika tebakan-tebakan awal tidak cukup dekat ke akar-akar sejati. Penggambaran kurva masing-masing persamaan secara grafik dapat membantu pemilihan tebakan awal yang bagus.

BAB III
PENUTUP

Kesimpulan

Metode Pencarian Akar

Dalam metode numerik, pencarian akar $f(x) = 0$ dilakukan secara lelaran (iteratif). Sampai saat ini sudah banyak ditemukan metode pencarian akar. Secara umum semua metode pencarian akar tersebut dapat dikelompokkan menjadi 2 golongan besar yaitu

a) Metode Tertutup Atau Metode Pengurung (*bracketing method*)
b) Metode terbuka

a) Metode Tertutup Atau Metode Pengurung (*bracketing method*)

Metode tertutup memerlukan selang $[a, b]$ yang mengandung akar. Sebagaimana namanya, selang tersebut "mengurung" akar sejati. Strategi yang dipakai adalah mengurangi lebar selang secara sistematis sehingga lebar selang tersebut semakin sempit dan karenanya menuju akar yang benar.
Ada dua metode klasik yang termasuk ke dalam metode tertutup, yaitu

a) metode bagi dua
b) metode regula-falsi

b) Metode Terbuka

Metode ini tidak memerlukan selang yang mengurung akar. Yang diperlukan hanyalah sebuah tebakan awal akar atau dua buah tebakan yang tidak perlu mengurung akar. Oleh karena itulah metodenya dinamakan metode terbuka.

Yang termasuk dalam metode terbuka adalah :

1. Metode lelaran titik tetap (*fixed-point interation*)
2. Metode Newton-Raphson
3. Metode *secant* .

DAFTAR ISI

1. D. Conte Samuel, Carl D. Boor. “ *Dasar-dasarAnalisaNumerik “* : Mc GrawHill . 1980.
2. C.Chapra Steven, P.C. Raymond. *“METODE NUMERIK,jilid 1”* : Mc GrawHill .1988.
3. Munir Rinaldi,. “ *Metode Numerik* “ Jakarta : Penerbit Erlangga . 2003
4. Munir Rinaldi,. *“ Bahan Kuliah 1F4058 Topik Khusus Informatika “*. ITB.
5. Curtis F.Gerald, Patrick O.Wheatley,. “ *Applied Numerical Anaysis*, 3rdEd “ . Pearson Education Inc . 2004
6.