Computatio: Journal of Computer Science and Information Systems, 7/1 (2023), 37 - 49

APLIKASI PENYELESAIAN NUMERIK PENCARIAN AKAR PERSAMAAN NON-LINIER DAN PENERAPANNYA DALAM MENYELESAIKAN ANALISIS *BREAK EVEN POINT*

Tri Sutrisno[1]
[1] Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Tarumanagara,
Jln. Letjen S. Parman No. 1, Jakarta, 11440, Indonesia
E-mail: [1]tris@fti.untar.ac.id,

Abstrak

Model matematika merupakan cara sederhana menerjemahkan suatu masalah ke dalam bahasa matematika dengan menggunakan persamaan, pertidaksamaan, atau fungsi. Persoalan yang melibatkan model matematika banyak muncul dalam berbagai disiplin ilmu pengetahuan, seperti dalam persoalan rekayasa di bidang ekonomi yaitu break even point yang terkadang berbentuk persamaan non-linier yang tidak ideal alias rumit berakibat tidak dapat diselesaikan dengan metode analitik sehingga solusi persoalan tersebut harus diselesaikan dengan metode numerik. Penyelesaian secara numerik melibatkan proses iterasi perhitungan berulang, jika dilakukan secara manual membutuhkan waktu relatif lama dan kemungkinan timbulnya nilai kesalahan (error) akibat manusia relatif besar. Dengan memanfaatkan teknologi informasi yang ada pada saat ini, dibuatlah suatu aplikasi penyelesaian numerik persamaan non-linier yang bertujuan untuk mendapatkan ketepatan dan kecepatan iterasi dalam perolehan hasil akhir, sehingga analisis numerik dapat dilakukan dengan lebih baik dan efektif. Metode yang digunakan pada penelitian ini adalah studi literatur, kuantitatif komparatif, simulasi numerik dan pembahasan pada setiap metode dilengkapi dengan algoritma, bagan alir (flow chart), dan program komputer dalam bahasa pemrograman C++. Hasil penelitian menunjukkan metode Newton Raphson paling efisien menyelesaikan analisis break even point, hal ini ditunjukkan dengan nilai galat terkecil yang diperoleh pada akhir proses iterasi dan jumlah iterasinya paling sedikit juga dibandingkan dengan metode yang lain.

Kata kunci *Metode Numerik,* Akar Persamaan *Non-linier, Break Even*

Abstract

Mathematical models are a simple way of translating a problem into mathematical language by using equations, inequalities, or functions. Many problems involving mathematical models arise in various scientific disciplines, such as in engineering problems in the economics field, namely break even points which are sometimes in the form of non-linear equations which are not ideal or complicated which results in not being able to be solved by analytical methods so that the solution to the problem must be solved by numerical method. Solving numerically involves an iterative process of repeated calculations, if done manually it takes a relatively long time and the possibility of human-induced errors is also relatively large. By utilizing existing information technology, an application for solving numerical non-linear equations is made which aims to obtain the accuracy and speed of iterations in obtaining the final result, so that numerical analysis can be carried out better and more effectively. The methods used in this research are literature study, comparative quantitative, numerical simulation and discussion of each method equipped

with algorithms, flow charts, and computer programs in the C++ programming language. The results showed that the Newton Raphson method was the most efficient in completing break even point analysis, this was indicated by the smallest error value obtained at the end of the iteration process and the least number of iterations compared to other methods.

Keywords *Numerical Methods, Roots of Non-linier Equations, Break Even Points*

1. PENDAHULUAN

Model matematika merupakan suatu cara sederhana untuk menerjemahkan suatu masalah sehari-hari ke dalam bahasa matematika dengan menggunakan persamaan, pertidaksamaan, atau fungsi. Persoalan yang melibatkan model matematika banyak muncul dalam berbagai disiplin ilmu pengetahuan, seperti pada persoalan rekayasa [1]. Untuk mendapatkan penyelesaian matematika yang menjabarkan model suatu persoalan nyata bidang rekayasa, sering solusi yang dicari berupa suatu nilai variabel x sedemikian rupa sehingga terpenuhi persamaan $f(x) = 0$ yang digunakan dalam model. Dalam beberapa kasus, melalui faktorisasi $f(x) = 0$ dapat diperoleh penyelesaian seperti yang diinginkan, akan tetapi lebih banyak jabaran persamaan dalam model mempunyai bentuk yang rumit, sehingga metode analitik matematika murni tidak dapat memberikan solusi [2]. Bila metode analitik tidak dapat lagi diterapkan, maka solusi persoalan sebenarnya masih dapat dicari dengan menggunakan metode numerik. Metode numerik adalah teknik yang digunakan untuk memformulasikan persoalan matematik sehingga dapat dipecahkan dengan operasi perhitungan/aritmetika biasa (penjumlahan, perkalian, pembagian, plus membuat perbandinga). Penyelesaian secara numerik umumnya melibatkan proses iterasi, perhitungan berulang dari data numerik yang ada. Jika proses iterasi tersebut dilakukan secara manual, akan membutuhkan waktu yang relatif lama dan kemungkinan timbulnya nilai kesalahan (*error*) akibat manusia itu sendiri juga relatif besar. Pada keadaan demikian ini aplikasi atau program komputer sangat dibutuhkan untuk mempercepat proses perhitungan tanpa membuat kesalahan. Selain mempercepat perhitungan numerik, dengan aplikasi program komputer dapat dicoba juga berbagai kemungkinan solusi yang terjadi akibat perubahan beberapa parameter tanpa menyita waktu dan pikiran. Solusi yang diperoleh juga dapat ditingkatkan ketelitiannya dengan mengubah-ubah nilai parameternya [1]. Pengaruh aplikasi program komputer dalam proses perhitungan numerik terhadap solusi persamaan non linier adalah menghasilkan kesalahan yang lebih kecil dan tingkat akurasi yang lebih akurat dari cara perhitungan yang lain [3].

Banyak masalah rekayasa dibidang ekonomi yang memerlukan penyelesaian secara numerik, salah satu diantaranya adalah analisis titik impas (*break even point*). Hal ini terlihat pada penelitian Nur Insani [4] yang menerapkan metode bagi dua (*bisection*) untuk menganalisis titik impas (*break even paoint*) dengan sebelumnya dikonversi ke suatu ukuran yang dapat dibandingkan dan akhirnya masalah tersebut direduksi menjadi masalah pencarian akar persamaan. Selanjutnya, Ismuniarto [5] melakukan penelitian lebih lanjut dengan membandingkan metode *bisection, regula falsi* dan *secant* dalam menyelesaikan analisis *break even point.* Penelitian tersebut dalam penghitungan untuk setiap iterasinya masih dilakukan secara manual sehingga membutuhkan waktu relatif lama dan rentang kemungkinan timbulnya nilai kesalahan (*error*) akibat manusia itu sendiri juga relatif besar. Hal inilah yang melatarbelakangi perlunya di buat suatu aplikasi penyelesaian numerik persamaan non-linier dan penerapannya dalam menyelesaiakan analisis *Break Even Point*. Adapun tujuan dari penelitian ini adalah untuk mendapatkan ketepatan dan kecepatan iterasi dalam perolehan hasil akhir, sehingga analisis numerik dapat dilakukan dengan lebih baik dan efektif. Selain itu juga untuk membandingkan metode *Bisection, Regula Falsi, Newton Raphson* dan *Secant* sehingga diketahui metode mana dengan ketelitian dan akurasi tinggi serta waktu penyelesaian yang singkat dalam menyelesaiakan analisis *Break Even Point*.

2. METODE PENELITIAN

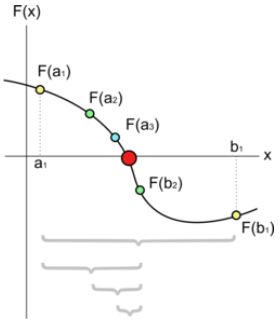
Metode yang digunakan dalam penelitian ini adalah studi literatur dengan melakukan penelusuran dan penelaahan terhadap beberapa literatur yang mempunyai relevansi dengan topik pembahasan, selanjutnya dilakukan identifikasi masalah dan kajian untuk memahami serta menentukan motivasi berdasarkan hasil studi literatur. Berdasarkan hasil identifikasi masalah dan motivasi yang mendorong dilakukannya penelitian diperoleh fokus penelitiannya adalah perancangan dan pembuatan aplikasi penyelesaian numerik pencarian akar persamaan non-linier dikerjakan dengan metode pengembangan perangkat lunak dalam melakukan desain sistem dan pembuatan aplikasi. Berdasarkan rancangan solusi yang dibuat, demonstrasi dibangun dengan tujuan menguji aplikasi yang dibuat untuk melihat kesesuaian rancangan dengan harapan yang ingin dicapai. Pengujian aplikasi dilakukan dengan melakukan *debugging* terhadap aplikasi menggunakan metode *blackbox* dan dievaluasi terhadap kelompok pengguna (*beta taster*) sebagai bagian dari evaluasi dengan pengguna sekaligus menilai pencapaian tujuan dari aplikasi. Analisis dilakukan terhadap hasil pengujian yang didapatkan dan laporan temuan penelitian berdasarkan data dan hasil analisis yang ada dibuat dan dilaporkan. Penelitian ini merupakan penelitian deskriptif dengan pendekatan kuantitatif untuk menggambarkan pembuatan program aplikasi penyelesaian numerik pencarian akar persamaan non linier. Pada Penelitian ini membahas dan membandingkan 4 metode penyelesaian numerik pencarian akar persamaan yaitu metode *Bisection, Regula Falsi*, *Newton Raphson* dan *Secant* untuk mendapatkan penyelesaian dengan ketelitian dan akurasi tinggi serta waktu penyelesaian singkat.

3. HASIL DAN PEMBAHASAN

Dalam metode numerik, pencarian akar $f(x) = 0$ dilakukan secara lelaran (iteratif). Sampai saat ini sudah banyak ditemukan metode pencarian akar, tetapi secara umum dapat dikelompokkan menjadi dua yaitu metode tertutup dan metode terbuka. Metode tertutup merupakan metode pencarian akar yang memerlukan interval tertutup $[a, b]$ yang memuat akar. Strategi yang dipakai mengurangi lebar interval secara sistematis sehingga lebar interval tersebut semakin sempit, dan akhirnya diperoleh akar persamaan. Syarat cukup keberadaan akar adalah jika $f(a) * f(b) < 0$ maka paling sedikit terdapat satu buah akar persamaan $f(x) = 0$ dalam interval $[a, b]$. Ada dua metode klasik yang termasuk kedalam metode tertutup, yaitu metode bagi dua (*bisection*) dan metode posisi palsu (*regula-falsi*). Masing masing metode dibahas lebih rinci di bawah ini.

Metode Bagi Dua *(Bisection)*

Tahap pertama dengan menetapkan sembarang nilai a dan b sebagai batas inverval nilai fungsi yang dicari, selanjutnya disubtitusikan ke fungsi untuk mendapatkan nilai fungsi $f(a)$ dan $f(b)$ serta periksa apakah $f(a) * f(b) < 0$. Apabila terpenuhi kondisi tersebut berarti terdapat akar fungsi dalam interval tinjauan, jika tidak demikian kembali harus menetapkan nilai a dan b sedemikian sehingga terpenuhi ketentuan $f(a) * f(b) < 0$. Selanjutnya membagi interval nilai fungsi yang dicari menjadi dua bagian $m = \frac{a+b}{2}$ dan periksa apakah nilai mutlak $f(m)$ <toleransi (batas simpangan kesalahan), jika memenuhi maka nilai $x = m$ merupakan solusi yang dicari. Jika tidak terpenuhi, ditetapkan batas interval baru dengan mengganti nilai $b = m$ jika $f(a) * f(m) < 0$ dan mengganti $a = m$ jika $f(a) * f(m) > 0$. Interval yang baru dibagi dua lagi dengan cara yang sama, begitu seterusnya sampai ukuran interval yang baru sangat kecil. Kondisi berhenti lelaran dapat dipilih salah satu dari kriteria berikut diantaranya lebar interval baru lebih kecil dari toleransi atau nilai fungsi di hampiran sama dengan nol atau lebih kecil dari epsilon mesin atau lebar interval baru lebih kecil dari pada nilai toleransi.

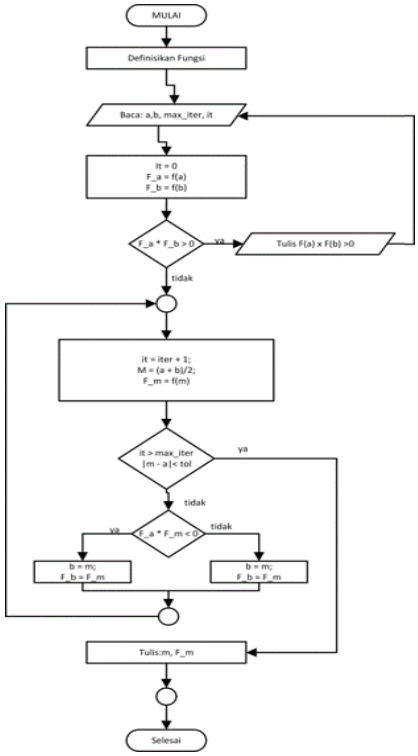


Gambar 1. Metode Bagi Dua

Alur pemikiran pencarian akar dengan metode Bagi dua (*Bisection*) dapat dituliskan dalam bentuk Algoritma program sebagai berikut.

a. Tentukan nilai a batas atas dan b batas atas interval, toleransi dan jumlah iterasi maksimum
b. Periksa apakah $f(a) * f(b) > 0$; jika ya, kembali untuk input nilai a dan b sedemikian sehingga $f(a) * f(b) < 0$.
c. Hitung nilai $m = \frac{a+b}{2}$
d. Jika nilai mutlak $|b - a| <$ toleransi, tuliskan m sebagai hasil perhitungan dan akhiri program; jika tidak, lanjutkan ke langkah berikutnya
e. Jika jumlah iterasi $>$ iterasi maksimum, akhiri program
f. Jika $f(a) * f(b) < 0$, maka $b = m$; jika tidak $a = m$
g. Kembali ke langkah (c)

Selanjutnya, untuk memudahkan dalam membuat program aplikasinya maka algoritma tersebut dituliskan dalam bentuk *flowchart* atau diagram alir yang memperlihatkan tahapan dari suatu program dan hubungan antar proses beserta pernyataannya.



Gambar 2. *Flowchart* Algoritma Metode Bagi Dua (*Bisection*)

Lebih lanjut, implementasi algoritma metode Bagi dua (*Bisection*) dengan menggunakan Bahasa Pemrograman C++ diberikan sebagai berikut.

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <cmath>
#include <iomanip>
using namespace std;
float f(float x);
int main() {
float a,b,m,error1,error2,tol;
int Iter_Maks;
int i=1;
cout<<"========METODE Bisection=======”<<endl<<endl;;
cout<<"f(x) =(((-1400*pow(1.2, x))/(pow(1.2, x)-1))-((150*x)/(pow(1.2, x)-1))+3750)”<<endl;
//User menginput nilai x1 dan x2
cout<<"Masukkan batas bawah (a): “; cin>>a;
cout<<"Masukkan batas atas (b): “;cin>>b;
if (f(a)*f(b)>0){
do{
   cout<<"Tidak terdapat akar pada interval tersebut, input lagi batas bawah dan batas atas”<<endl;
   cout<<"Masukkan batas bawah (a): “; cin>>a;
   cout<<"Masukkan batas atas (b): “;cin>>b;
   }
while (f(a)*f(b)>0);
}
cout<<"Toleransi: “;cin>>tol;
cout<<"Jumlah Iterasi Maksimum: “;cin>>Iter_Maks;
{
cout<<setw(5)<<"I”<<setw(15)<<"a”<<setw(20)<<"b”<<setw(20)<<"m”<<setw(20)<<"f(a)”;
cout<<setw(20)<<"f(b)”<<setw(20)<<"f(m)”<<setw(20)<<"error1”<<setw(20)<<"error2”<<endl;
do
{
m=(a+b)/2;
error1= abs(a-m);
error2= abs(b-m);
cout<<setw(5)<<i<<setw(15)<<a<<setw(20)<<b<<setw(20)<<m<<setw(20)<<f(a)<<setw(20);
cout<<f(b)<<setw(20)<<f(m)<<setw(20)<<error1<<setw(20)<<error2<<endl;
if(f(a)*f(m)<0)
{b=m;}
else
{a=m;
}
i++;
}
while( (error1 > tol) && (error2 > tol) && (I <= Iter_Maks));
```

```
cout<<"Approx. root = "<<m<<endl;
cout<<"Banyaknya iterasi : "<<i-1;
}
getch();
return 0;
}
float f(float x) {
return  (((-1400*pow(1.2, x))/(pow(1.2, x)-1))-((150*x)/(pow(1.2, x)-1))+3750);
}
```

Selanjutnya hasil eksekusi Program Aplikasi Metode Bagi Dua (*Bisection*) sebagai berikut.

```
=======METODE Bisection=======

f(x) =(((-1400*pow(1.2, x))/(pow(1.2, x)-1))-((150*x)/(pow(1.2, x)-1))+3750)
Masukkan batas bawah (a): 2
Masukkan batas atas (b): 10
Toleransi: 0.0000001
Jumlah Iterasi Maksimum: 50
```

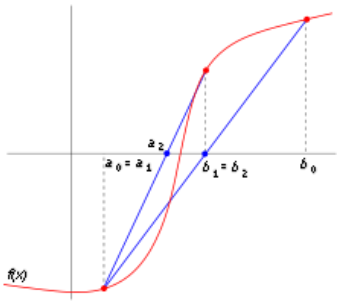
i	a	b	m	f(a)	f(b)	f(m)	error1	error2
1	2	10	6	-1513.64	1791.42	1191.88	4	4
2	2	6	4	-1513.64	1191.88	487.109	2	2
3	2	4	3	-1513.64	487.109	-191.209	1	1
4	3	4	3.5	-191.209	487.109	194.174	0.5	0.5
5	3	3.5	3.25	-191.209	194.174	15.6767	0.25	0.25
6	3	3.25	3.125	-191.209	15.6767	-83.7954	0.125	0.125
7	3.125	3.25	3.1875	-83.7954	15.6767	-33.1246	0.0625	0.0625
8	3.1875	3.25	3.21875	-33.1246	15.6767	-8.49702	0.03125	0.03125
9	3.21875	3.25	3.23438	-8.49702	15.6767	3.64577	0.015625	0.015625
10	3.21875	3.23438	3.22656	-8.49702	3.64577	-2.41154	0.0078125	0.0078125
11	3.22656	3.23438	3.23047	-2.41154	3.64577	0.620623	0.00390625	0.00390625
12	3.22656	3.23047	3.22852	-2.41154	0.620623	-0.894581	0.00195312	0.00195312
13	3.22852	3.23047	3.22949	-0.894581	0.620623	-0.13676	0.000976562	0.000976562
14	3.22949	3.23047	3.22998	-0.13676	0.620623	0.241986	0.000488281	0.000488281
15	3.22949	3.22998	3.22974	-0.13676	0.241986	0.0526267	0.000244141	0.000244141
16	3.22949	3.22974	3.22961	-0.13676	0.0526267	-0.0420632	0.00012207	0.00012207
17	3.22961	3.22974	3.22968	-0.0420632	0.0526267	0.00528263	6.10352e-005	6.10352e-005
18	3.22961	3.22968	3.22964	-0.0420632	0.00528263	-0.0183901	3.05176e-005	3.05176e-005
19	3.22964	3.22968	3.22966	-0.0183901	0.00528263	-0.00655367	1.52588e-005	1.52588e-005
20	3.22966	3.22968	3.22967	-0.00655367	0.00528263	-0.000616479	7.62939e-006	7.62939e-006
21	3.22967	3.22968	3.22967	-0.000616479	0.00528263	0.00233308	3.8147e-006	3.8147e-006
22	3.22967	3.22967	3.22967	-0.000616479	0.00233308	0.000839272	1.90735e-006	1.90735e-006
23	3.22967	3.22967	3.22967	-0.000616479	0.000839272	0.000111396	9.53674e-007	9.53674e-007
24	3.22967	3.22967	3.22967	-0.000616479	0.000111396	-0.00027157	4.76837e-007	4.76837e-007
25	3.22967	3.22967	3.22967	-0.00027157	0.000111396	-8.00868e-005	2.38419e-007	2.38419e-007
26	3.22967	3.22967	3.22967	-8.00868e-005	0.000111396	0.000111396	2.38419e-007	0

```
Approx. root = 3.22967
Banyaknya iterasi : 26
```

Gambar 3. Hasil Eksekusi Program Metode Bagi Dua (*Bisection*)

Metode Posisi Palsu *(Regula Falsi)*

Metode posisi palsu (Regula Falsi) merupakan perbaikan dari metode bagi dua (*bisection*) yang kecepatan konvergensinya sangat lambat. Kecepatan konvergensi dapat ditingkatkan bila nilai $f(a)$ dan $f(b)$ turut diperhitungkan. Oleh karena itu 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 seolah-olah berlaku menggantikan kurva $f(x)$ dan memberikan posisi palsu dari akar.



Gambar 4. Metode Regulafalsi

Gradien garis AB = gradien garis BC

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

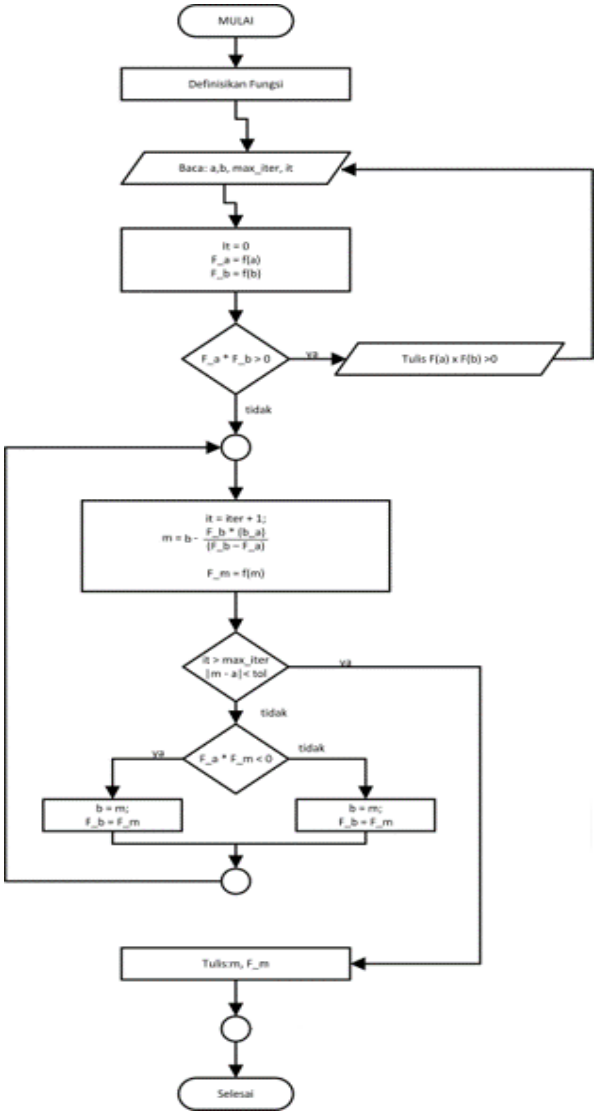
yang dapat disederhanakan menjadi

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

Alur pemikiran pencarian akar dengan metode posisi palsu (*regula falsi*) dapat dituliskan dalam bentuk Algoritma program sebagai berikut.

a. Tentukan nilai a batas atas dan b batas atas interval, toleransi dan jumlah iterasi maksimum
b. Periksa apakah $f(a) * f(b) > 0$; jika ya, kembali untuk input nilai a dan b sedemikian sehingga $f(a) * f(b) < 0$.
c. Hitung nilai $m = b - \frac{f(b)(b-a)}{f(b)-f(a)}$
d. Jika nilai mutlak $|b - a| <$ toleransi, tuliskan m sebagai hasil perhitungan dan akhiri program; jika tidak, lanjutkan ke langkah berikutnya
e. Jika jumlah iterasi $>$ iterasi maksimum, akhiri program
f. Jika $f(a) * f(b) < 0$, maka $b = m$; jika tidak $a = m$
g. Kembali ke langkah (c)

Selanjutnya, untuk memudahkan dalam membuat program aplikasinya maka algoritma tersebut dituliskan dalam bentuk *flowchart* atau diagram alir yang memperlihatkan tahapan dari suatu program dan hubungan antar proses beserta pernyataannya.



Gambar 5. *Flowchart* Algoritma Metode Posisi Palsu (*Regula Falsi*)

Algoritma regula falsi hampir sama dengan algoritma bagidua kecuali pada perhitungan nilai ***m***. Lebih lanjut, implementasi algoritma Metode Posisi Palsu (*Regula Falsi*) dengan menggunakan Bahasa Pemrograman C++ dan hasil eksekusi Program Aplikasi Metode Posisi Palsu (*Regula Falsi*) sebagai berikut.

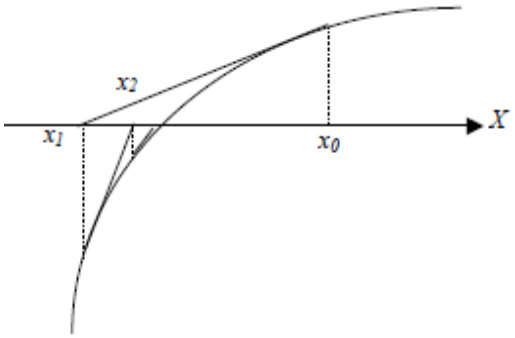
```
========METODE Regula Falsi=======

f(x) =(((-1400*pow(1.2, x))/(pow(1.2, x)-1))-((150*x)/(pow(1.2, x)-1))+3750)
Masukkan batas bawah (a): 2
Masukkan batas atas (b): 10
Toleransi: 0.0000001
Jumlah Iterasi Maksimum: 50
   i          a          b          m          f(a)          f(b)          f(m)          error1          error2
   1          2         10    5.66381      -1513.64       1791.42       1106.08         3.66381         4.33619
   2          2    5.66381     4.1169      -1513.64       1106.08        545.841          2.1169         1.54691
   3          2     4.1169    3.55584      -1513.64       545.841        230.764         1.55584         0.56106
   4          2    3.55584    3.35002      -1513.64       230.764        90.1428         1.35002        0.205819
   5          2    3.35002    3.27414      -1513.64       90.1428        34.0469         1.27414       0.0758798
   6          2    3.27414    3.24611      -1513.64       34.0469        12.6915         1.24611       0.0280292
   7          2    3.24611    3.23575      -1513.64       12.6915        4.70726         1.23575       0.0103617
   8          2    3.23575    3.23192      -1513.64       4.70726        1.74275         1.23192      0.00383115
   9          2    3.23192     3.2305      -1513.64       1.74275       0.644666          1.2305      0.00141692
  10          2     3.2305    3.22998      -1513.64      0.644666       0.238463         1.22998      0.000523806
  11          2    3.22998    3.22978      -1513.64      0.238463      0.0883252         1.22978      0.000193596
  12          2    3.22978    3.22971      -1513.64     0.0883252      0.0324788         1.22971      7.20024e-005
  13          2    3.22971    3.22968      -1513.64     0.0324788      0.0119476         1.22968      2.64645e-005
  14          2    3.22968    3.22967      -1513.64     0.0119476     0.00436327         1.22967      9.77516e-006
  15          2    3.22967    3.22967      -1513.64     0.00436327     0.00156715         1.22967      3.57628e-006
  16          2    3.22967    3.22967      -1513.64     0.00156715    0.000456306         1.22967      1.43051e-006
  17          2    3.22967    3.22967      -1513.64    0.000456306    0.000111396         1.22967      4.76837e-007
  18          2    3.22967    3.22967      -1513.64    0.000111396   -8.00868e-005         1.22967      2.38419e-007
  19    3.22967    3.22967    3.22967   -8.00868e-005    0.000111396   -8.00868e-005               0      2.38419e-007
Approx. root = 3.22967
Banyaknya iterasi : 19
```

Gambar 6. Hasil Eksekusi Program Metode Posisi Palsu (*Regula Falsi*)

Metode Newton Raphson

Newton Raphson merupakan metode penyelesaian persamaan non linier dengan menggunakan pendekatan satu titik awal dan mendekatinya dengan memperhatikan slope atau gradien. Titik pendekatan dinyatakan dengan persamaan $x_{baru} = x_0 - \frac{f(x_0)}{f'(0)}$ dengan $f'(x_0) \neq 0$ dan ilustrasi dapat dilihat pada gambar berikut.

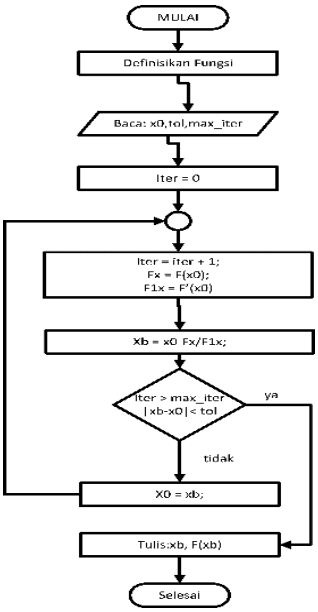


Gambar. 7 Metode Newton Raphson

Alur pemikiran pencarian akar dengan metode Newton Raphson dapat dituliskan dalam bentuk Algoritma program sebagai berikut.

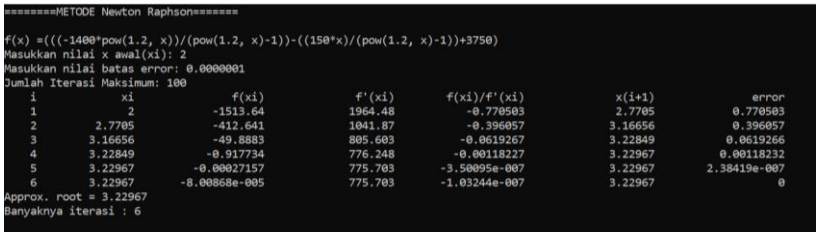
a. Tentukan tebakan awal x_r, toleransi dan jumlah iterasi maksimum
b. Hitung nilai $x_{baru} = x_0 - \frac{f(x_0)}{f'(x_0)}$
c. Jika nilai mutlak Jika nilai mutlak $(x_{baru} - x_0) < toleransi$, tulis x_{baru} sebagai hasil perhitungan, dan akhiri program; jika tidak, lanjutkan ke langkah berikutnya
d. Jika jumlah iterasi>iterasi maksimum, akhiri program
e. $x_0 = x_{baru}$ dan kembali ke langkah (b)

Selanjutnya, untuk memudahkan dalam membuat program aplikasinya maka algoritma tersebut dituliskan dalam bentuk *flowchart* atau diagram alir yang memperlihatkan tahapan dari suatu program dan hubungan antar proses beserta pernyataannya.



Gambar 8. *Flowchart* Algoritma Metode *Newton Raphson*

Lebih lanjut, implementasi algoritma metode Newton Raphson dengan menggunakan Bahasa Pemrograman C++ diberikan sebagai berikut. Selanjutnya hasil eksekusi Program Aplikasi *Newton Raphson* sebagai berikut.



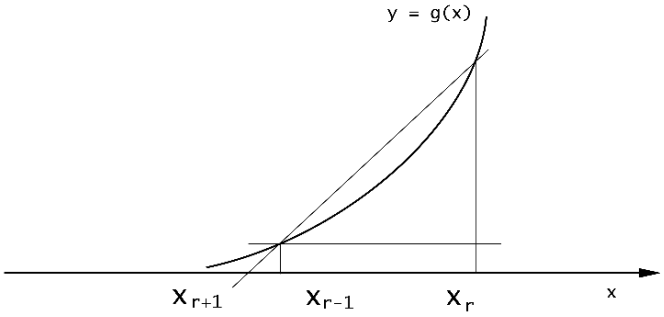
```
=======METODE Newton Raphson=======

f(x) =(((-1400*pow(1.2, x))/(pow(1.2, x)-1))-((150*x)/(pow(1.2, x)-1))+3750)
Masukkan nilai x awal(xi): 2
Masukkan nilai batas error: 0.0000001
Jumlah Iterasi Maksimum: 100
   i            xi            f(xi)          f'(xi)       f(xi)/f'(xi)          x(i+1)           error
   1             2          -1513.64         1964.48          -0.770503          2.7705        0.770503
   2        2.7705          -412.641         1041.87          -0.396057         3.16656        0.396057
   3       3.16656          -49.8883         805.603          -0.0619267         3.22849       0.0619266
   4       3.22849          -0.917734         776.248         -0.00118227         3.22967       0.00118232
   5       3.22967        -0.00027157         775.703       -3.50095e-007         3.22967      2.38419e-007
   6       3.22967       -8.00868e-005         775.703       -1.03244e-007         3.22967                0
Approx. root = 3.22967
Banyaknya iterasi : 6
```

Gambar 9. Hasil Eksekusi Program *Newton Raphson*

Metode *Secant*

Prosedur lelaran metode *Newton Raphson* memerlukan perhitungan turunan fungsi $f'(x)$, padahal tidak semua fungsi mudah dicari turunannya sehingga perlu dicari bentuk lain yang ekuivalen. Modifikasi metode Newton Raphson dinamakan metode secant.



Gambar. 10 Metode Newton Raphson

Berdasarkan gambar dapat dihitung gradien

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

Selanjutnya subtitusikan dalam rumus Newton-Raphson $x_{baru} = x_1 - \frac{f(x_0)}{f'(x_0)}$ sehingga diperoleh $x_{baru} = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$ yang merupakan prosedur lelaran metode secant. Sepintas metode secant mirip dengan metode regula falsi, namum sesungguhnya prinsip dasar keduanya berbeda yang dirangkum pada table berikut.

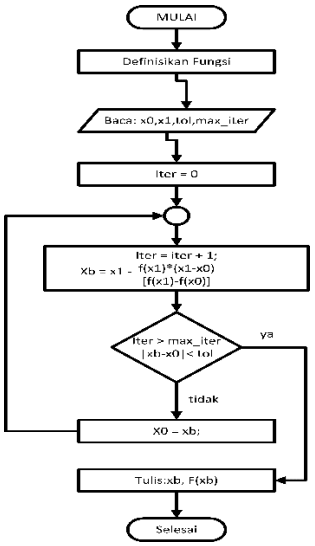
Tabel 1 Perbandingan Metode Regula Falsi dan Secant

No	Metode Regula Falsi	Metode Secant
1	Diperlukan dua buah nilai awal a dan b (ujung-ujung interval) sedemikian sehingga $f(a)f(b) < 0$	Diperlukan dua buah nilai awal x_0 dan x_1 (tebakan awal akar) tetapi tidak harus $f(x_0)f(x_1) < 0$
2	Pada lelaran pertama tidak ada perbedaan antara regula falsi dan secant. Perbedaan muncul pada lelaran kedua perpotongan garis lurus dengan sumbu-x tetap berada didalam interval yang memuat akar.	Pada lelaran pertama tidak ada perbedaan antara regula falsi dan secant. Perbedaan muncul pada lelaran kedua perpotongan garis lurus dengan sumbu-x mungkin menjauhi akar.
3	Berdasarkan nomor 2 diatas, lelarannya selalu konvergen	Berdasarkan nomor 2 diatas, lelarannya mungkin divergen

Alur pemikiran pencarian akar dengan metode secant dapat dituliskan dalam bentuk Algoritma program sebagai berikut.

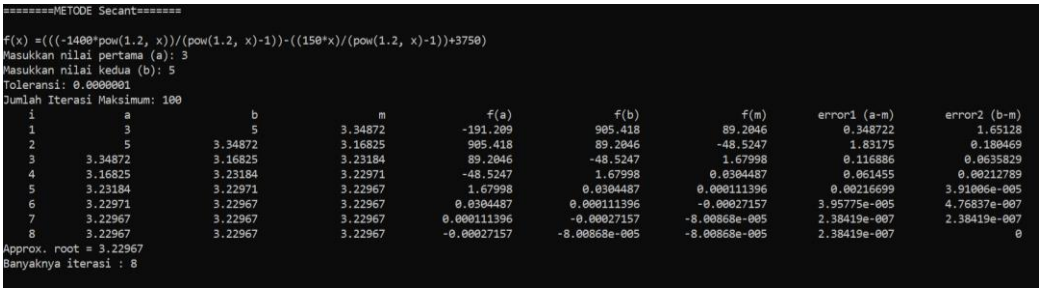
a. Tentukan nilai awal x_0 dan nilai kedua x_1, toleransi dan jumlah iterasi maksimum
b. Hitung nilai $x_{baru} = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$
c. Jika nilai mutlak $|x_{baru} - x_1| <$ toleransi, tuliskan x_{baru} sebagai hasil perhitungan dan akhiri program; jika tidak, lanjutkan ke langkah berikutnya
d. Jika jumlah iterasi $>$ iterasi maksimum, akhiri program
e. $x_0 = x_{baru}$ kembali ke Langkah langkah (b)

Selanjutnya, untuk memudahkan dalam membuat program aplikasinya maka algoritma tersebut dituliskan dalam bentuk *flowchart* atau diagram alir yang memperlihatkan tahapan dari suatu program dan hubungan antar proses beserta pernyataannya.



Gambar 11. *Flowchart* Algoritma Metode Secant

Lebih lanjut, implementasi algoritma Secant dengan menggunakan Bahasa Pemrograman C++ dan hasil eksekusi Program Aplikasi Metode Secant sebagai berikut.



```
========METODE Secant=======

f(x) =(((-1400*pow(1.2, x))/(pow(1.2, x)-1))-((150*x)/(pow(1.2, x)-1))+3750)
Masukkan nilai pertama (a): 3
Masukkan nilai kedua (b): 5
Toleransi: 0.0000001
Jumlah Iterasi Maksimum: 100
   i            a              b              m            f(a)            f(b)            f(m)        error1 (a-m)        error2 (b-m)
   1            3              5        3.34872        -191.209         905.418         89.2046            0.348722            1.65128
   2            5        3.34872        3.16825         905.418         89.2046        -48.5247             1.83175            0.180469
   3      3.34872        3.16825        3.23184         89.2046        -48.5247         1.67998            0.116886           0.0635829
   4      3.16825        3.23184        3.22971        -48.5247         1.67998        0.0304487           0.061455           0.00212789
   5      3.23184        3.22971        3.22967         1.67998        0.0304487      0.000111396          0.00216699          3.91006e-005
   6      3.22971        3.22967        3.22967        0.0304487      0.000111396       -0.00027157          3.95775e-005          4.76837e-007
   7      3.22967        3.22967        3.22967      0.000111396       -0.00027157      -8.00868e-005          2.38419e-007          2.38419e-007
   8      3.22967        3.22967        3.22967       -0.00027157      -8.00868e-005      -8.00868e-005          2.38419e-007                     0
Approx. root = 3.22967
Banyaknya iterasi : 8
```

Gambar 12. Hasil Eksekusi Program Metode Secant

Analisis *Break Even Point*

Break Event Point merupakan kondisi dimana total pendapatan sama dengan total biaya, atau dengan kata lain titik dimana keuntungan sama dengan nol, sehingga Analisa *Break Even Point* sangat diperlukan bagi manajemen dalam pengambilan keputusan untuk mengetahui hubungan antara biaya, volume dan keuntungan. Sebagai contoh dalam pertimbangan pengambilan keputusan pemilihan pembelian komputer Pentium atau AMD yang taksiran biaya dan keuntungan untuk setiap computer diberikan dalam tabel berikut.

Tabel 2 Biaya dan Keuntungan untuk 2 Komputer sendiri

No		Komputer Pentium	Komputer AMD
1	Biaya Pembelian ($)	-3000	-10000
2	Bertambahnya Biaya Perawatan/thn, $	-200	-50
3	Keuntungan dan Kenikmatan tahunan $/thn	1000	4000

Dalam hal ini tanda negatif menunjukkan biaya atau kerugian dan tanda positif menunjukkan keuntungan, supaya dapat dipertimbangkan maka biaya harus di transformasi ke ukuran yang dapat diperbandingkan. Misalnya, biaya pembelian awal dapat transformasi serangkaian pembayaran tahunan dengan rumus $A_P = P\frac{i(1+i)^n}{(1+i)^n+1}$ dengan A_p besarnya pembayaran tahunan, P biaya pembelian, i tingkat bunga dan n banyaknya tahun. Selanjutnya seiring berjalan waktu dibutuhkan biaya perawatan yang dapat dihitung dengan deret hitung gradient rumus $A_m = G\left[\frac{1}{i} - \frac{n}{(1+i)^n-1}\right]$ dengan A_m biaya pemeliharaan, G pertambahan perawatan, i tingkat bunga, dan n banyaknya tahun. Dalam hal ini diasumsikan dana merupakan pinjaman dengan bunga 20% sedemikian sehingga diperoleh harga total untuk komputer Pentium $A_t = -3000\frac{0.20(1+0.20)^n}{(1+0.20)^n+1} - 200\left[\frac{1}{0.20} - \frac{n}{(1+0.20)^n-1}\right] + 1000$ selanjutnya dapat disederhanakan menjadi $A_t = \frac{-600(1+0.20)^n}{(1.2)^n+1} + \frac{200n}{1.2^n-1}$ dan untuk komputer AMD $A_t = \frac{-2000(1.2)^n}{(1.2)^n+1} + \frac{50n}{1.2^n-1} + 3750$ sehingga diperoleh kesamaan $\frac{-600(1+0.20)^n}{(1.2)^n+1} + \frac{200n}{1.2^n-1} = \frac{-2000(1.2)^n}{(1.2)^n+1} + \frac{50n}{1.2^n-1} + 3750$ lebih lanjut disederhanakan menjadi satu ruas persamaan diperoleh $f(n) = \frac{-1400(1.2)^n}{(1.2)^n-1} - \frac{150n}{(1.2)^n-1} + 3750$ dan turunan fungsi tersebut diperoleh $f'(n) = \frac{1400(1.2)^n \ln(1.2) - 150(1.2)^n(1-n\ln(1.2)) + 150}{((1.2)^n-1)^2}$ yang selanjutnya turunan fungsi tersebut digunakan dalam metode Newton Raphson.

Persamaan model titik impas mempunyai bentuk yang rumit, sehingga metode analitik matematika murni tidak dapat memberikan solusi. Bila metode analitik tidak dapat lagi digunakan, maka salah satu solusi yang dapat digunakan adalah dengan metode numerik. Ada

beberapa metode dalam pencarian akar persamaan dengan metode numerik diantaranya adalah metode bagi dua (*bisection*), metode posisi palsu (*regula falsi*) dan metode *secant*. Selanjutnya dengan tiga metode tersebut akan digunakan untuk menyelesaikan pencarian akar atau pembuat nol dari persamaan $f(n) = \frac{-1400(1.2)^n}{(1.2)^n-1} - \frac{150n}{(1.2)^n-1} + 3750$. Selanjutnya untuk perbandingan hal yang dibandingkan adalah kecepatan proses tiap metode dengan melakukan 10 kali percobaan untuk nilai tebakan awal yang berbeda (nilai a atau b yang tertulis tebal merupakan tebakan awal untuk metode *Newton Raphson*) dengan nilai toleransi 0.0000001. Hasil perhitungan dapat dilihat pada tabel berikut.

Tabel 3 Hasil Pengujian Perbandingan Kecepatan Iterasi

No	a	b	*Bisection*	*Regula Falsi*	*Newton Raphson*	*Secant*
1	**3**	5	24	8	4	8
2	**4**	1	25	36	5	7
3	**1**	5	25	42	8	8
4	1	**70**	29	51	5	28
5	3	**40**	28	9	5	8
6	**5**	1	25	39	7	9
7	4	**2**	24	16	6	8
8	3	**7**	25	8	8	8
9	4	**3**	23	8	5	5
10	3	**9**	26	9	6	8
			Jumlah = 254	**Jumlah = 226**	**Jumlah = 59**	**Jumlah = 97**

Dari hasil tabel menjelaskan bahwa untuk analisis *break even* metode *Newton Raphson* merupakan metode yang paling cepat dalam melakukan proses iterasi, ini terlihat dari jumlah keseluruhan iterasi untuk ke10 percobaan yang dilakukan. Hasil menunjukkan untuk metode *Bisection* memiliki 254 jumlah iterasi, metode *Regula falsi* 226 jumlah iterasi, metode *Newton Raphson* 59 jumlah iterasi dan metode *Secant* 97 jumlah iterasi.

4. KESIMPULAN

Aplikasi penyelesaian numerik pencarian akar persamaan non-linier yang dibuat dengan bahasa pemrograman C++ mengeluarkan hasil berupa nilai akar pendekatan yang diperoleh dengan nilai fungsinya pada setiap iterasi sedemikian sehingga sampai lebih kecil dari nilai toleransi yang ditentukan serta nilai error pada setiap iterasi. Selanjutnya dalam pengujian dilakukan 10 kali percobaan pada setiap metode dengan nilai toleransi 0.0000001 diperoleh nilai akar-akar persamaan yang sama, meskipun demikian terdapat perbedaan jumlah iterasi yang dibutuhkan masing-masing metode, metode *Bisection* 254 jumlah iterasi, metode *Regula Falsi* 226 jumlah iterasi, metode *Newton Raphson* 59 jumlah iterasi dan metode *Secant* 97 jumlah iterasi menunjukkan bahwa metode *Newton Raphson* merupakan metode yang paling efisien dalam melakukan analisis *Break Even Point*. Lebih lanjut, aplikasi ini juga dapat digunakan untuk menyelesaikan persamaan non-linier lain, baik fungsi aljabar maupun fungsi transenden dengan mengganti definisi $f(x)$ di awal program.

UCAPAN TERIMA KASIH

Ucapan terimakasih ditujukan kepada kepada Lembaga Penelitian dan Pengabdian kepada Masyarakat Universitas Tarumangara (LPPM UNTAR) yang telah memberikan dukungan dan pendanaan, sehingga dapat terselenggaranya kegiatan penelitian ini.

DAFTAR PUSTAKA

[1] Munir, Rinaldi, 2015, *Metode Numerik: Revisi Keempat,* Ed. 1, Informatika Bandung.

[2] Nasution, A dan Zakaria, H., 2011, *Metode Numerik dalam Ilmu Rekayasa Sipil,* Ed. 2, ITB Bandung.

[3] Yahya dan Nur, A, M., 2018, Pengaruh Aplikasi C# dalam Proses Perhitungan Numerik Terhadap Solusi Persamaan Non Linier, *Jurnal Informatika dan Teknologi,* No. 2, Vol. 1, hal. 79-87.

[4] Insani, Nur, 20006, Penerapan Metode Bagi Dua (Bisection) Pada Analisis Pulang Pokok (Break Even), *Prosiding Seminar Nasional MIPA 2006,* Yogyakarta, 3 Februari.

[5] Ismuniarto, 2016, Perbandingan Metode Pengapitan Akar (Bisection, Regula Falsi, Secant) Persamaan Non Linier dalam Menyelesaikan Analisis Break Even, *Skripsi,* Fakultas Sains dan Teknologi, Universitas Islam Negeri Alauddin, Makassar.