SISTEM PERSAMAAN NON LINEAR

METODE BISEKSI

Oleh

Muh. Firdaus

F1D020054

1. Tujuan

Tujuan dari dilakukan pratikum ini adalah sebagai berikut:

a. Untuk mengetahui metode biseksi dalam menyelesaikan persamaan non linear.
b. Untuk mencari titik potong dari dua fungsi persamaan non linear.
c. Dapat mengetahui pengaruh banyaknya iterasi, ukuran interval, kesalahan toleransi dalam penyelesaian persamaan non linear.

2. Percobaan

Percobaan mencari titik potong pada persamaan non linear antara fungsi $f_1(x) = x^2-2x+1$ dan $f_2(x) = x^3-(x+2)e^{-2x}+1$ dapat dilakukan dengan metode biseksi. Langkah pertama untuk mencari titik potong dari dua fungsi tersebut yaitu dengan mendefinisikan fungsi *f(x)* yang akan dicari akarnya. Jika fungsi $f_1(x)$ dan $f_2(x)$ disamakan nilainya, maka didapatkan fungsi baru $f(x) = x^2-x^3-2x+x*exp(-2*x)+2*exp(-2*x)$. Setelah mendefinisikan fungsi *f(x)*, kemudian menentukan batas atas dan bawah untuk fungsi tersebut serta kesalahan toleransinya. Langkah selanjutnya dapat disimulasikan ke dalam suatu algoritma untuk menghitung akar dari fungsi diatas sebagai berikut:

a. Mendefinisikan fungsi *f(x)* yang akan dicari akarnya.
b. Tentukan interval *a* dan *b.*
c. Tentukan kesalahan toleransi (*eS*).
d. Hitung *f(a)* dan *f(b).*
e. Jika *f(a)*f(b)>0*, maka proses diberhentikan kerena tidak mengandung akar pada interval tersebut, jika tidak maka lanjutkan.
f. Hitung nilai *x = (a+b)/2* dan *f(x).*
g. Jika *f(x)*f(a)<0*, maka *b=x* dan *f(b)=f(x)*, jika tidak *a=x* dan *f(a)=f(x).*
h. Jika *|f(x)|<eS*, maka proses berhenti dan akarnya adalah *x*, jika tidak maka ulangi langkah f.

Jika algoritma diatas diaplikasikan kedelam program Java menjadi sebagai berikut:

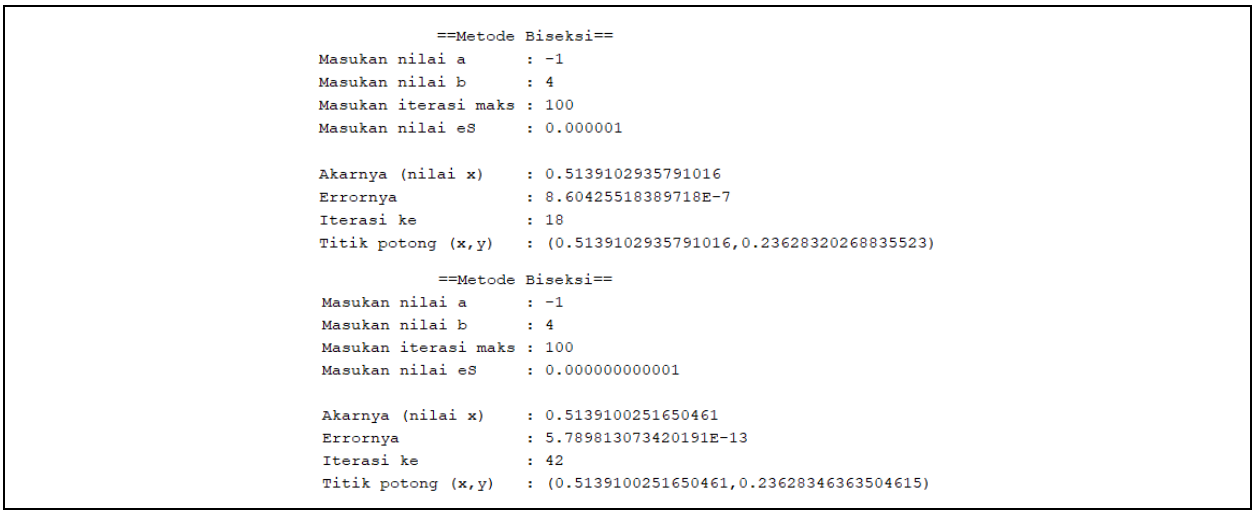
```
public class MetnumMandiri {
    static double fx(double x){
        double y = x*x-x*x*x-2*x+x*exp(-2*x)+2*exp(-2*x);
        return y;
    }

    static double fx1(double x){
        double y = x*x-2*x+1;
        return y;
    }

    public static void main(String[] args) {
        double a, b, eS, fa, fb;
        int max_iterasi;

        Scanner input = new Scanner (System.in);
        System.out.println("\t    ==Metode Biseksi== ");
        System.out.print("Masukan nilai a      : ");
        a = input.nextDouble();
        System.out.print("Masukan nilai b      : ");
```

```
        b = input.nextDouble();
        System.out.print("Masukan iterasi maks : ");
        max_iterasi = input.nextInt();
        System.out.print("Masukan nilai eS     : ");
        eS = input.nextDouble();

        double x = 0.0;
        double fx = 10.0;
        fa = fx(a);
        fb = fx(b);

        if ((fa*fb)>0){
            System.out.println("Tidak Ada Akar");
        }else {
            int i;
            //System.out.println("|Iterasi ke-  | a | b | x | fa | fb | error |");
            for ( i = 0; i < max_iterasi; i++) {
                x = (a+b)/2;
                fx = fx(x);
                if(abs(fx)>eS){
                    if ((fx*fa)<0){
                        b = x;
                        fb = fx;
                    }else {
                        a = x;
                        fa = fx;
                    }
                }else{
                    break;
                }
                //System.out.println("| "+i+"          | "+a+" | "+b+" | "+x+" |"+fa+"
                | "+fb+" | "+fx+" | ");
            }
            double y = fx1(x);

            System.out.println("\nAkarnya (nilai x)    : "+x);
            System.out.println("Errornya             : "+abs(fx));
            System.out.println("Iterasi ke           : "+j);
            System.out.println("Titik potong (x,y)   : ("+x+","+y+")");
        }
    }
}
```

3. Hasil Percobaan dan Pembahasan

Dari *source code* di atas, dibuat dua buah *function,* masing-masing *function* untuk menghitung *f(x)* dan $f_1(x)$. Selain dari dua *function* tersebut, terdapat *main function* untuk mengeksekusi program. Di dalamnya terdapat perintah untuk memasukan nilai batas awal, batas akhir, iterasi maksimum dan kesalahan toleransi. Selain itu, di dalam *main function* juga terdapat *if statement* dan *while looping* untuk menentukan kondisi berhenti dari proses perhitungan dan iterasi sampai nilai *error*-nya kurang dari kesalahan toleransi sehingga didapatkan suatu akar yang memenuhi fungsi *f(x)*.

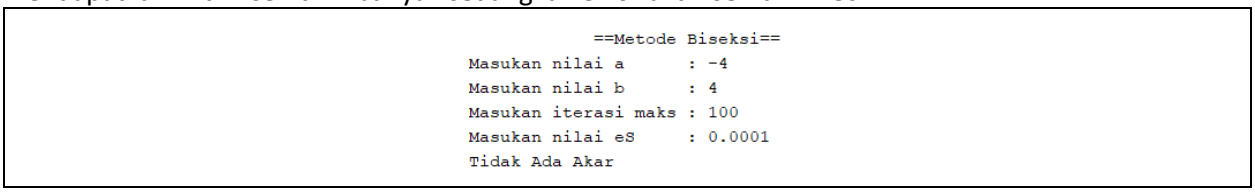
```
                ==Metode Biseksi==
Masukan nilai a      : -1
Masukan nilai b      : 4
Masukan iterasi maks : 100
Masukan nilai eS     : 0.0001

Akarnya (nilai x)    : 0.5139007568359375
Errornya             : 2.9710494080736893E-5
Iterasi ke           : 15
Titik potong (x,y)   : (0.5139007568359375,0.23629247420467436)
```



Gambar 3.1 Tiga percobaan (eS sama)

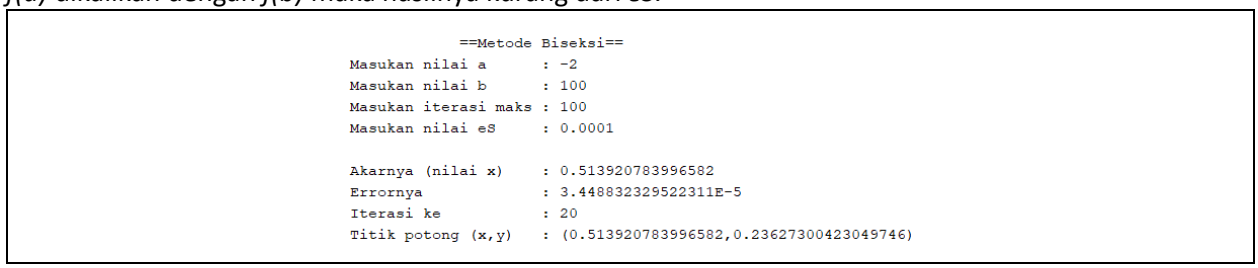
Percobaan pada **Gambar 3.1** dilakukan dengan memberikan masukan batas awal (a), batas atas (b), dan iterasi maksimum dengan nilai yang sama pada tiga kali percobaan yaitu a = -1, b = 4 dan iterasi maksimum = 100. Jika diberikan nilai *eS* = 10^{-4} maka akar dari fungsi didapatkan pada iterasi ke 15 dan *error* = 2,971E-5, kemudian apabila nilai *eS* dikecilkan menjadi *eS* = 10^{-6} maka didapatkan akar pada iterasi ke 18 dan *error* = 8,6042E-7, sedangkan apabila nilai eS dikecilkan menjadi eS = 10^{-12} maka didapatkan akar pada iterasi ke 42 dan *error* = 5,7898E-13. Dari ketiga percoba tersebut dengan nilai *eS* berbeda beda didapatkan akar yang sama yaitu x = 0,5139, kemudian nilai x ini dimasukan ke dalam fungsi $f_1(x) = x^2-2x+1$ sehingga didapatkan nilai y adalah 0,2362. Dengan demikian titik potong (x , y) antara dua fungsi yaitu (0,5140 , 2,362).

Dari percobaan pada **Gambar 3.1** didapatkan bahwa semakin kecil nilai *eS* maka iterasi untuk mendapatkan nilai x semakin banyak sedangkan *error* akan semakin kecil.



Gambar 3.2 Range diubah (a=-4)

Percobaan pada **Gambar 3.2** dilakukan dengan mengubah *range* [a,b] dimana nilai a = -4 dan b = 4, sementara untuk iterasi maksimum = 1000 dan *eS* = 10^{-4} tidak terdapat akar. Hal ini terjadi karena nilai *f(a)* dikalikan dengan *f(b)* maka hasilnya kurang dari *eS.*



Gambar 3.3 Range diubah (a=-2;b=100)

Percobaan pada **Gambar 3.3** dilakukan dengan mengubah masukan batas awal (a), batas atas (b) dan iterasi maksimum dengan nilai yang sama pada tiga kali percobaan yaitu a = -2, b = 100, iterasi maksimum = 100 dan *eS* = 10^{-4} maka didapatkan nilai akar x = 0,5139 pada iterasi ke-20 dan *error* = 3,4488E-15.

4. Kesimpulan

a. Dalam menyelesaikan persamaan non linear dapat menggunakan metode biseksi. Metode biseksi merupakan metode untuk mencari akar dari suatu persamaan non linear dengan menentukan interval yang dimana didalamnya terdapat akar yang akan dicari. Interval akan terus diperkecil dengan dibagi dua hingga didapatkan *error* kurang dari kesalahan toleransi.
b. Titik potong dari dua persamaan non linear diperoleh dengan menyamakan dua persamaan sehingga menghasilkan suatu persamaan baru. Persamaan baru ini kemudian dicari akarnya (nilai x) menggunakan metode biseksi, kemudian akar yang didapatkan dimasukan ke dalam salah satu persamaan non linear sebelum disamakan nilanya sehingga menghasilkan nilai y. Nilai x dan y merupakan titik potong dari dua persaman non linear tersebut.
c. Metode biseksi sangat berpengaruh terhadap nilai *eS,* semakin kecil/teliti nilai *eS*-nya semakin banyak iterasi yang dilakukan. Selain nilai *eS,* faktor lain yang menyebabkan iterasi semakin banyak adalah ukuran interval.

Daftar Pustaka

[1] Sutarno, H. dan Rachmatin, D., “Metode Numerik Dengan Pendekatan Algoritmik”, Bandung: PT Sniar Baru Algensindo, 2005.
[2] Sutarno, H. dan Rachmatin, D., “Hands-Out Metode Numerik”, Bandung: Universitas Pendidikan Indonesia, 2008.
[3] Vulandari, R. T., “Metode Numerik: Teori, Kasus, dan Aplikasi”, Surabaya: Mavendra Pers, 2017.
[4] Luknanto, J., “Metoda Numerik”, Bahan kuliah Metoda Numerik Teknik Sipil UGM, Jogyakarta, 2001.
[5] Atmika, A. I K., “Diktat Metode Numerik”, Denpasar: Universitas Udayana, 2016.