Laporan Tugas 2

Kelompok B09



Analisis Numerik



Anggota Kelompok:

1. Gilang Gumilar – 1506737792
2. Gunawan – 1506757926
3. Sarah Dewi Fadlia – 1506688853
4. Stefanus James – 1506738076



# Daftar Isi

[**Daftar Isi**](#_7gnpo616246q) **2**

[**Latar Belakang**](#_23t5xtb27igt) **4**

[**Implementasi**](#_e0dfhx7kmjjc) **5**

[Bisection Method](#_1997vwttgkne) 5

[Fixed Point Method](#_mtz81shh9530) 7

[Newton’s Method](#_upbe7ytldyx3) 9

[Modified Newton Method](#_f9j1ty39zxsf) 9

Modified Newton for Non-linear System 10

[**Hasil Eksperimen**](#_qecml3go9lr7) **13**

[Bisection Method](#_bvvg7zx9os2g) 13

[Fixed Point Method](#_2qgvop6w1xkl) 13

[Newton’s Method](#_lny9dmcx991y) 13

[Modified Newton](#_h0orbqtiyx) 14

Modified Newton for Non-linear System 17

[**Analisis**](#_8n82d4srgjiw) **19**

[Bisection Method](#_g27r2y694gq9) 19

[Kaitan algoritma dengan teori](#_su27h1m8n0ox) 19

[Kelebihan algoritma](#_z83ywggg24u) 19

[Kekurangan algoritma](#_xzejodh7fgqy) 19

[Akurasi](#_7d0g9999hsf6) 19

[Fixed Point Method](#_g7pr4vvlvqjn) 20

[Kaitan algoritma dengan teori](#_xtkovhukmlj2) 20

[Kelebihan algoritma](#_byrjygp2jd3p) 20

[Kekurangan algoritma](#_7p3rt5kpnq80) 20

[Akurasi](#_9bolcvrqmc9b) 20

[Newton Method](#_730v68uw9ig7) 20

[Kaitan algoritma dengan teori](#_3bdkxvtj1u4) 20

[Kelebihan algoritma](#_jl429tl5mqdw) 20

[Kekurangan algoritma](#_pwbwyz91lmtu) 20

[Akurasi](#_asakvzw8wigd) 21

[Modified Newton Method](#_3t2m6gpgwwtm) 21

[Kaitan algoritma dengan teori](#_pl2rf88dl8kj) 21

[Kelebihan algoritma](#_hiwkzd93an67) 21

[Kekurangan algoritma](#_cl757hevbgi7) 21

[Akurasi](#_1a9lugafgth) 21

**Kesimpulan 22**

**Kontribusi** **23**

[**Daftar Pustaka**](#_gfvedm6sdsxx) **2**4

# Latar Belakang

**Algoritma bisection**

Diketahui fungsi f(x) dan interval [a,b]. Kita tahu bahwa f(x) berubah tanda pada [a,b], hal ini berarti f(a) dan f(b) memiliki tanda yang berbeda. Tugas kita adalah mencari titik c pada interval [a,b] dimana c berada dalam interval [a,b] dan f(c) = 0.

Ide dari metode bisection adalah metode yang digunakan untuk melokalisasi solusi x\* dengan cara membagi dua interval di mana x\* berada secara terus menerus hingga ukuran interval tersebut cukup kecil dan hanya memuat x\*. Solusi yang diberikan adalah interval tertutup [a,b] di mana f berubah tanda. Kita membagi interval menjadi 2 dan f harus berubah tanda, baik di kiri maupun di kanan (f(a)•f(b) < 0, sign function dari a dan b berbeda). Kemudian kita mengganti [a,b] dengan interval baru dari interval awal yang tadi telah dibagi 2 di mana f berubah tanda. Proses ini diulangi hingga ukuran interval tersebut cukup kecil dan hanya memuat x\*.

**Algoritma fixed-point**

Metode titik tetap adalah metode yang mencari titik tetap dari sebuah *iterated function g(x)*. Titik tetap itu sendiri adalah *x* yang memenuhi *g(x)* = *x*. Metode ini dapat membantu menyelesaikan persamaan nonlinear *f(x)* = 0. Solusi yang diberikan adalah modifikasi persamaan *f(x)* = 0 sehingga salah satu ruasnya menjadi hanya terdapat satu variabel saja. Contoh *f(x)* = *x*2 - 2*x* - 3 = 0 menjadi *g(x) = √(2x + 3)*, kemudian lakukan skema iterasi *xk+1 = g(xk)* hingga konvergen.

**Algoritma Newton**

Newton Raphson method atau Newton’s method digunakan untuk mencari akar dari persamaan non-linear dengan mendekatkannya ke persamaan linear. Ide dasarnya adalah sebuah persamaan nonlinear memiliki sifat linear secara lokal (sebuah titik). Oleh karena itu persamaan nonlinear didekatkan dengan persamaan linear pada titik tersebut. Solusi yang diberikan dari newton method ini adalah dengan mendekatkan persamaan nonlinear ke persamaan linear. Dengan menggunakan deret Taylor yang hanya sampai menggunakan turunan pertama maka fungsi yang didapat dari deret Taylor itu sama dengan fungsi garis singgung pada suatu titik x. Selanjutnya, kita harus menebak nilai x pertama yang nantinya akan diiterasi sehingga nilai x konvergen pada suatu titik (yang nantinya merupakan akar dari persamaan nonlinear).

**Algoritma Modified Newton**

Ide dasar algoritma newton: Secara lokal suatu persamaan nonliner memiliki sifat linear. Cara mendapatkan solusinya adalah dengan mendekati fungsi non-linear tersebut dengan fungsi linear (garis singgung) di suatu titik. Metode newton akan semakin berkurang kinerjanya terhadap percarian solusi non-linear yang memiliki akar dengan multiplisitas lebih dari 1. Untuk itu modifikasi metode newton diperlukan kecepatan konvergensi dapat tetap efisien. Memodifiakasi terhadap metode newton dapat bervariasi. Dalam tugas ini akan digunakan pemanfaatan turunan pertama dan kedua suatu persamaan dalam memodifikasi metode newton.

# Implementasi

### **Bisection Method**

Berikut ini merupakan flowchart dari jalannya algoritma Bisection method.

Sumber gambar: <http://www.maths.qmul.ac.uk/~wj/MTH5110/notes/notes05.pdf>

Fungsi bisection bertujuan untuk melokalisasi solusi x\* untuk memperoleh solusi dari persamaan non-linear. Fungsi ini menerima input nilai interval [a,b], persamaan non-linear yang akan dicari solusinya, serta nilai toleransi.

Fungsi ini bekerja dengan membagi interval [a,b] menjadi 2 untuk mendapatkan nilai tengah dari interval tersebut, proses ini berlangsung secara iteratif. Kemudian akan dicek pula apakah f(nilai tengah) jika dikalikan dengan f(interval bawah) lebih dari atau kurang dari 0.

Apabila f(interval bawah) dikali f (nilai tengah) lebih dari nol, maka nilai interval bawah akan di-set sama dengan nilai tengah, hal ini karena berarti kemungkinan solusi akar persamaan non-linearnya ada di samping kanan dari nilai tengah, maka kita memperkecil interval untuk memperoleh solusi dengan memindahkan nilai interval bawah ke nilai tengah.

Sebaliknya, jika nilai dari f(interval bawah) dikali dengan f(nilai tengah) kurang dari nol, berarti kemungkinan solusi akar persamaan non-linearnya ada di kiri dari nilai tengah, sehingga kita perlu set (memindahkan) nilai interval atas sama dengan nilai tengah, hal ini agar kita dapat memperkecil interval dan memperoleh solusi.

Iterasi akan dilakukan selama hasil dari interval atas dikurangi interval bawah lebih dari nilai toleransi. Output dari fungsi adalah akar (nilai x) dari persamaan non-linear yang diberikan.

erikut ini merupakan source code dari algoritma bisection method

|  |
| --- |
| **function [x, error, iter, time, tol] = bisection(a,b,f,TOL = 10^-12)    % init error, iter, time, m = mean  error = 10^6;  iter = 0;  time = cputime;  m = 0;    % init f(a) and f(b)  fa = f(a);  fb = f(b);    % check if f(a)\*f(b) < 0  if(fa \* fb > 0)  disp('Function has same sign at both endpoints.')  endif    % loop until b-a > tol  while (abs(b - a) > TOL)  % find the middle point of the interval  m = (a + b) / 2;  fm = f(m);  % move the interval, so it becomes closer to the root  if (fa \* fm > 0)   a = m;  else  b = m;  endif   iter++;  endwhile  % output : root (x), error, num of iteration, running time  disp('-----------------------------------------')  disp(' output')  disp('-----------------------------------------')  x = m;  result = f(m);  error = abs(f(m))  iter  time = cputime - time  endfunction** |

### **Fixed Point Method**

Berikut ini disajikan 3 variasi g(x) untuk digunakan dalam metode titik tetap:

1. 𝑓(𝑥) = 𝑥3 + 2𝑥2 + 10𝑥 − 20 = 0
   1. 𝑔1(𝑥) = (-𝑥3 - 2𝑥2 + 20) / 10
   2. 𝑔2(𝑥) = √((-𝑥3 - 10𝑥 + 20) / 2)
   3. 𝑔3(𝑥) = 20 / (𝑥2 - 2𝑥 + 10) **(konvergen)**
2. 𝑓(𝑥) = 𝑥x = 2
   1. 𝑔1(𝑥) = 1 / log2(𝑥)
   2. 𝑔2(𝑥) = 𝑥√2 **(konvergen)**
   3. 𝑔3(𝑥) = log𝑥(2)
3. 10𝑒−𝑥sin(2𝜋𝑥) − 2 = 0
   1. 𝑔1(𝑥) = -ln(2) + ln(10) + ln(sin(2𝜋𝑥))
   2. 𝑔2(𝑥) = arcsin(2 / (10𝑒−𝑥)) / (2𝜋) **(konvergen)**
   3. 𝑔3(𝑥) = -log10(2) + 1 + log10(sin(2𝜋𝑥))
4. 𝑥4 − 6𝑥3 + 12𝑥2 − 10𝑥 + 3 = 0
   1. 𝑔1(𝑥) = (𝑥4 − 6𝑥3 + 12𝑥2 + 3) / 10 **(konvergen)**
   2. 𝑔2(𝑥) = √((-𝑥4 + 6𝑥3 + 10𝑥 - 3) / 12) **(konvergen)**
   3. 𝑔3(𝑥) = ∜(6𝑥3 - 12𝑥2 + 10𝑥 - 3) **(konvergen)**
5. 𝑥2 − cos(𝜋𝑥) = 0
   1. 𝑔1(𝑥) = cos(𝜋𝑥) / 𝑥
   2. 𝑔2(𝑥) = arccos(𝑥2) / 𝜋 **(konvergen)**
   3. 𝑔3(𝑥) = √(1 / sec(𝜋𝑥))

Banyak yang tidak konvergen karena tidak memenuhi syarat g(x) agar konvergen untuk metode titik tetap yaitu, |g’(c)| < 1, minimal untuk c disekitar solusi.

|  |
| --- |
| function [x, i, error, time] = fixed\_point(x0, f, g, TOL = 10^-12)  % fprintf(' i g(x)\n')  % fprintf('----------- -----------\n')    tic;    xs = zeros(1, 1);  xs(1) = x0;    diff = TOL + 1;  i = 1;    while (diff > TOL && i < 1000)  xs(i + 1) = g(xs(i)); % update x using g  diff = abs(xs(i + 1) - xs(i));  % fprintf('%12.0f %12.8f\n', i, xs(i + 1));  i += 1;  endwhile    x = xs(i);  error = abs(f(x));    time = toc; endfunction |

### **Newton’s Method**

|  |
| --- |
| **function [x, iter, diff, time] = newton(x0, f ,fp ,tol)  t0 = clock();  i = 1;  x(1) = x0;  diff = 10^6;    while(i < 1000 && diff > tol && x != 0 && fp(x(i)) != 0)  x(i+1) = x(i) - (f(x(i))/fp(x(i)));  diff = abs(x(i+1) - x(i));  i = i+1;  endwhile    x = x(i);  iter = i;  time = etime(clock(), t0); end** |

Algoritma newton di atas mengambil input berupa tebakan awal x, persamaan nonlinear f, turunan pertama persamaan nonlinear f’, dan toleransi. Output berupa akar dari persamaan nonlinear, banyaknya iterasi untuk mendapatkan akar tersebut, waktu komputasi, dan selisih antara nilai akar dengan nilai x sebelumnya.

Di dalam program terdapat iterasi yang dibatasi sampai 999 kali. Iterasi akan berhenti bila sudah melebihi 999 kali iterasi, selisih yang lebih besar daripada batas toleransi, jika nilai x = 0, dan jika turunan pertama persamaan nonlinear yang disubsitusikan nilai x(i) = 0.

Setiap iterasi akan diupdate nilai selisih dan nilai x.

### **Modified Newton Method**

Berikut implementasi algoritma Modified Newton.

|  |
| --- |
| **function [x, iter, error, time] = newton(xo, f, fp, fpp, TOL = 10^-5)**  **% xo nilai awal x**  **% f persamaan non-linear f(x) = 0**  **% fp turunan pertama persamaan f**  **% fpp turunan kedua persamaan f**  **% TOL toleransi perbedaan nilai x disetiap iterasi**  **time = cputime;  i = 1;  error = 10^6;  diff = 10^6;    x = zeros(1);  x(1) = xo;    while(i < 1000 && diff > TOL && f(x(i)) != 0) % limit iteration to 1000  x(i + 1) = x(i) - f(x(i))\*fp(x(i)) / ((fp(x(i)))^2 - f(x(i))\*fpp(x(i)));  diff = abs(x(i + 1) - x(i));  i += 1;  endwhile   x = x(i);  time = cputime - time;  iter = i;  error = diff; end** |

Metode modified newton, muncul sebagai solusi dalam mengatasi permasalahan penurunan kecepatan konvergensi algoritma Newton biasa dalam menyelesaikan persoalan persamaan non-linear yang memiliki akar dengan multiplisitas lebih dari 1. Dengan memodifikasi fungsi pengubah x pada setiap iterasinya, akan didapatkan hasil konvergensi yang lebih optimal dibanding metode newton biasa untuk persoalan persamaan dengan akar yang memiliki multiplisitas lebih dari 1.

### **Modified Newton’s Method for Non-linear System**

𝑓(𝑥, 𝑦) = 𝑥2 + 3𝑥𝑦2+ 8𝑦 − 100 = 0

𝑓(𝑥, 𝑦) = cos(𝜋𝑥) − 𝑥𝑦2 + 35 = 0

**Matrix F**

|  |  |
| --- | --- |
| F(x, y) | 𝑥2 + 3𝑥𝑦2 + 8𝑦 − 100 |
| cos(𝜋𝑥) − 𝑥𝑦2 + 35 |

**Matrix jacobian**

|  |  |  |
| --- | --- | --- |
| J = | 2x + 3y2 | 6xy + 8 |
| -πsin(πx) -y2 | -2xy |

**Implementasi Algoritma dengan bantuan octave**

|  |
| --- |
| **function F = jm(x, y)**  **% fungsi untuk mengembalikan matrix F(x0) c11 = 2\*x + 3\*y^2; c12 = 6\*x\*y + 8; c21 = -pi\* sin(pi\*x) - y^2; c22 = -2\*x\*y; F = [c11 c12; c21 c22]; end** |

|  |
| --- |
| **function J = jm(x, y)**  **% fungsi untuk mengembalikan Jacobian matrix c11 = 2\*x + 3\*y^2; c12 = 6\*x\*y + 8; c21 = -pi\* sin(pi\*x) - y^2; c22 = -2\*x\*y; J = [c11 c12; c21 c22]; end** |

|  |
| --- |
| **function res = nl\_modified\_newton(x0, y0)  iter = 1;  TOL = 10^-6;    x = zeros(2, 1);  x(:, 1) = [x0; y0];    while (iter < 1000)  J = jm(x(1, iter), x(2, iter));    [L, U] = lu(J); % pada step ini Jacobian Matrix yang telah dihitung dipakai 2 - 3 iterasi. Untuk menekan biaya komputasi.  for (i=1:3)  F = ff(x(1, iter), x(2, iter));  y = pinv(L)\*(-F);  d = pinv(U)\*y;   x(:, iter + 1) = x(:, iter) + d; %3.b  if (abs(x(:, iter + 1) - x(:, iter)) < TOL) % 4  break;  endif  iter += 1;  endfor   endwhile  res = x end** |

# Hasil Eksperimen

#### **Bisection Method**

Hasil percobaan terlampir. Berikut beberapa hasil eksperimen.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **TOL = 10^-6** | | | | **TOL = 10^-4** | | | |
|  | **a = -2, b = 1.7** | | | | **a = -2, b = 1.7** | | | |
| Soal | Solusi | Iterasi | Error | Waktu | Solusi | Iterasi | Error | Waktu |
| fa | 1.3688 | 22 | 1.30E-05 | 0 | 1.3688 | 16 | 9.36E-04 | 0 |
| fb | 1.5596 | 22 | 8.99E-08 | 0 | 1.5596 | 16 | 5.87E-05 | 0 |
| fc | 1.1029 | 22 | 5.68E-06 | 0 | 1.1030 | 16 | 4.06E-04 | 0 |
| fd | 1 | 22 | 0 | 0 | 0.99998 | 16 | 9.77E-15 | 0 |
| fe | -0.43843 | 22 | 3.09E-06 | 0 | -0.43844 | 16 | 4.23E-05 | 0 |
|  | **TOL = 10^-10** | | | | **TOL = 10^-12** | | | |
|  | **a = -1, b = 1.8** | | | | **a = -1, b = 1.8** | | | |
| Soal | Solusi | Iterasi | Error | Waktu | Solusi | Iterasi | Error | Waktu |
| fa | 1.3688 | 35 | 2.08E-10 | 0 | 1.3688 | 42 | 6.98E-12 | 0 |
| fb | 1.5596 | 35 | 7.39E-11 | 0 | 1.5596 | 42 | 1.48E-12 | 0 |
| fc | 0.033143 | 35 | 3.53E-09 | 0 | 0.033143 | 42 | 2.08E-11 | 0 |
| fd | 0.99999 | 35 | 0 | 0 | 0.99999 | 42 | 0 | 0 |
| fe | -0.43843 | 35 | 1.69E-10 | 0 | -0.43843 | 42 | 2.67E-13 | 0 |

#### **Fixed Point Method**

Hasil percobaan terlampir.

#### **Newton’s Method**

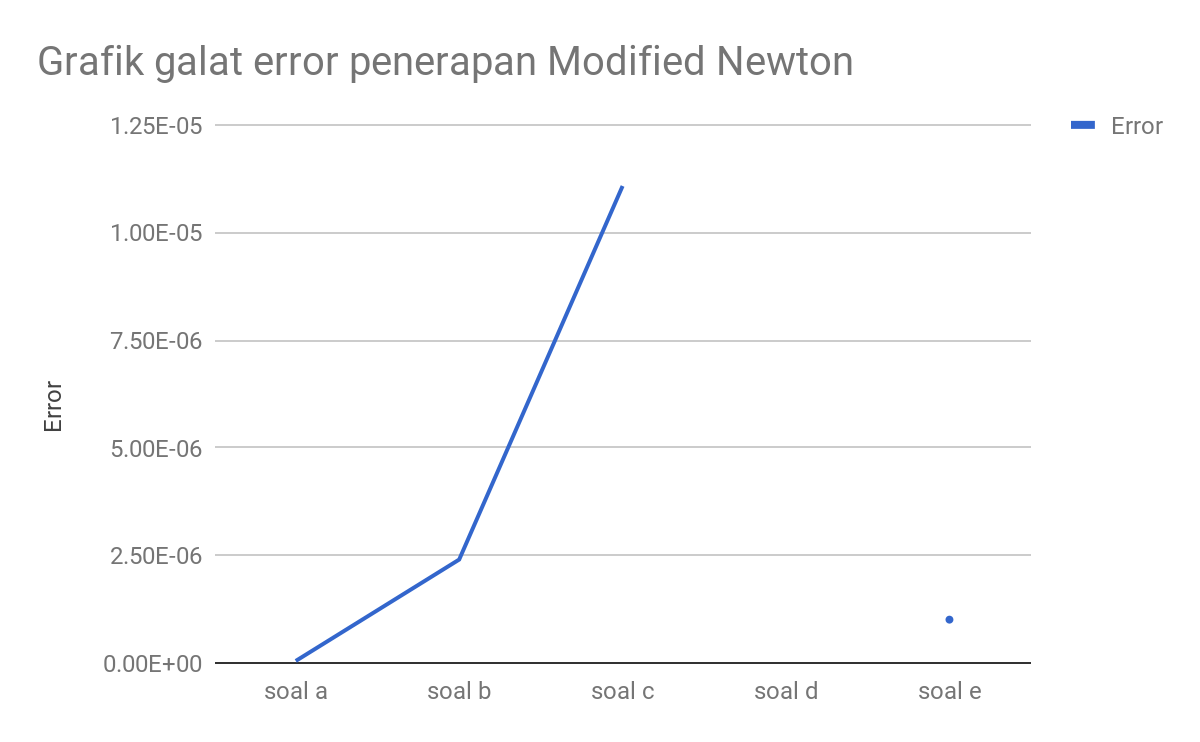
Hasil percobaan terlampir.

#### **Modified Newton**

Tabel dibawah ini merupakan sebagian dari, keseluruhan hasil percobaan terlampir.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **TOL = 10^-6** | | | | **TOL = 10^-6** | | | |
|  | **Xo = 1** | | | | **Xo = 2** | | | |
| Soal | Solusi | Iterasi | Error | Waktu | Solusi | Iterasi | Error | Waktu |
| fa | 1.368808108 | 5 | 4.68E-08 | 0 | 1.368808108 | 5 | 4.03E-06 | 0 |
| fb | 1.559610469 | 6 | 2.40E-06 | 0 | 1.559610469 | 6 | 1.00E-05 | 0 |
| fc | 1.102933997 | 4 | 1.11E-05 | 0 | 1.358165227 | 14 | 2.46E-06 | 0 |
| fd | 1 | 1 | 1000000 | 0 | 1.000000139 | 5 | 0.0009139382578 | 0 |
| fe | 0.4384307795 | 6 | 1.01E-06 | 0 | 0.4384307795 | 25 | 1.80E-07 | 0 |
|  | **TOL = 10^-6** | | | | **TOL = 10^-6** | | | |
|  | **Xo = 3** | | | | **Xo = 4** | | | |
| Soal | Solusi | Iterasi | Error | Waktu | Solusi | Iterasi | Error | Waktu |
| fa | 1.368808 | 6 | 2.53E-06 | 0 | 1.368808 | 6 | 9.66E-05 | 0 |
| fb | -0.124217660222088+0.06790025218606949i | 1000 | 1.36E-01 | 0 | NaN | 5 | NaN | 0 |
| fc | 0.449261 | 9 | 2.02E-06 | 0 | -140.000000 | 625 | 7.48E-06 | 0 |
| fd | 3.000000 | 1 | 1000000 | 0 | 2.500000 | 3 | 0 | 0 |
| fe | 0.438431 | 17 | 1.25E-08 | 0 | 0.438431 | 54 | 3.60E-07 | 0 |
|  | **TOL = 10^-6** | | | | TOL = 10^-6 | | | |
|  | **Xo = 5** | | | | Xo = 2 | | | |
| Soal | Solusi | Iterasi | Error | Waktu | Solusi | Iterasi | Error | Waktu |
| fa | 1.368808 | 6 | 2.02E-06 | 0 | 1.368808108 | 5 | 4.03E-06 | 0 |
| fb | NaN | 4 | NaN | 0 | 1.559610469 | 6 | 1.00E-05 | 0 |
| fc | -110.500000 | 8 | 1.07E-08 | 0 | 1.358165227 | 14 | 2.46E-06 | 0 |
| fd | 0.999963 | 7 | 5.13E-05 | 0 | 1.000000139 | 5 | 0.0009139382578 | 0 |
| fe | -5679.690468 | 1000 | 7.38E+03 | 0 | 0.4384307795 | 25 | 1.80E-07 | 0 |

#### **Chart**



#### 

#### **Modified Newton for Non-Linear System**

Tabel dibawah ini merupakan sebagian dari, keseluruhan hasil percobaan terlampir.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Hasil Eksperiment |  |  |  |  |  |  |  |
| Nilai Awal | (1, 1) |  |  |  |  |  |  |
| **Iterasi 1** | **Menghitung J dan faktorisasi L, U dari J** | | | | **Menyelesaikan Ly = -F(x) dan Ud = y** | | |
|  | J | 5 | 14 |  | F | -87 |  |
|  | -1 | -2 |  | 33 |  |
|  | L | 1 | 0 |  | y | 87 |  |
|  | -0.2 | 1 |  | -15.6 |  |
|  | U | 5 | 14 |  | d | 7.20E+01 |  |
|  | 0 | 0.8 |  | -1.95E+01 |  |
|  |  |  |  |  | **Iterasi: x1 = x0 + d** | |  |
|  |  |  |  |  | x1 | 7.30E+01 |  |
|  |  |  |  |  | -1.85E+01 |  |
|  |  |  |  |  |  |  |  |
| **Iterasi 2** |  |  |  |  | **Menyelesaikan Ly = -F(x) dan Ud = y** | | |
|  |  |  |  |  | F | 7.49E+04 |  |
|  |  |  |  |  | -2.50E+04 |  |
|  |  |  |  |  | y | -7.49E+04 |  |
|  |  |  |  |  | 9.98E+03 |  |
|  |  |  |  |  | d | -4.99E+04 |  |
|  |  |  |  |  | 1.25E+04 |  |
|  |  |  |  |  | **Iterasi: x1 = x0 + d** | |  |
|  |  |  |  |  | x1 | -4.98E+04 |  |
|  |  |  |  |  | 1.25E+04 |  |
|  |  |  |  |  |  |  |  |
| **Iterasi 3** |  |  |  |  | **Menyelesaikan Ly = -F(x) dan Ud = y** | | |
|  |  |  |  |  | F | -2.32E+13 |  |
|  |  |  |  |  | 7.73E+12 |  |
|  |  |  |  |  | y | 2.32E+13 |  |
|  |  |  |  |  | -3.09E+12 |  |
|  |  |  |  |  | d | 1.55E+13 |  |
|  |  |  |  |  | -3.87E+12 |  |
|  |  |  |  |  | Iterasi: x1 = x0 + d | |  |
|  |  |  |  |  | x1 | 1.55E+13 |  |
|  |  |  |  |  | -3.87E+12 |  |
|  |  |  |  |  |  |  |  |
| **Iterasi 4** | **Menghitung J dan faktorisasi L, U dari J** | | | | **Menyelesaikan Ly = -F(x) dan Ud = y** | | |
|  | J | 4.48E+25 | -3.59E+26 |  | F | 6.93E+38 |  |
|  | -1.49E+25 | 1.20E+26 |  | -2.31E+38 |  |
|  | L | 1 | 0 |  | y | -6.93E+38 |  |
|  | -0.33333 | 1 |  | 1.59E+26 |  |
|  | U | 4.48E+25 | -3.59E+26 |  | d | 6.47E+09 |  |
|  | 0.00E+00 | 8.24E+137 |  | 1.93E+12 |  |
|  |  |  |  |  | **Iterasi: x1 = x0 + d** | |  |
|  |  |  |  |  | x1 | 1.55E+13 |  |
|  |  |  |  |  | -1.93E+12 |  |
|  |  |  |  |  |  |  |  |
| **Iterasi 5** |  |  |  |  | **Menyelesaikan Ly = -F(x) dan Ud = y** | | |
|  |  |  |  |  | F | 1.73E+38 |  |
|  |  |  |  |  | -5.77E+37 |  |
|  |  |  |  |  | y | -1.73E+38 |  |
|  |  |  |  |  | 3.98E+25 |  |
|  |  |  |  |  | d | 2.32E+09 |  |
|  |  |  |  |  | 4.83E+11 |  |
|  |  |  |  |  | **Iterasi: x1 = x0 + d** | |  |
|  |  |  |  |  | x1 | 1.55E+13 |  |
|  |  |  |  |  | -1.45E+12 |  |

#### **Cara menentukan nilai a dan b bisection**

##### **Alasan penentuan**

Ambil nilai x secara random, kemudian masukkan nilai x ke dalam f(x) hingga f(x) berubah tanda (misalnya: dari negatif ke positif). Dalam menentukan nilai a dan b, f(a)•f(b) < 0 atau sign function di kedua ujung berbeda.

##### **Penjelasan langkah-langkahnya**

Dalam menentukan nilai a dan b, dapat dilakukan dengan mengambil sembarang nilai x, kemudian masukkan nilai x tersebut ke dalam f(x) untuk diuji kapan f(x) berganti tanda (misalnya: dari f(x) bernilai negatif berganti tanda menjadi f(x) bernilai positif). Jika f(x) telah berganti tanda, maka nilai x yang menyebabkan f(x) berganti tanda akan menjadi interval [a,b].

Misal:

f(x) = x3 - 4x - 9 = 0

x = 0, f(0) = -9

x = 1, f(1) = -12

x = 2, f(2) = -9

x = 3, f(3) = 6

f(x) berubah tanda ketika nilai x = 2 dan nilai x = 3 maka interval dari f(x) adalah [2,3].

Cek apakah f(a)•f(b) < 0? f(2)•f(3) = -9 • 6 = -54, -54 < 0. Syarat f(a)•f(b) < 0 terpenuhi.

#### **Cara menentukan nilai awal x fixed point**

##### **Alasan penentuan**

Agar metode ini konvergen (jika konvergen) dengan cepat, seharusnya *x* yang dipilih adalah x yang dekat dengan *x\*.* Tapi karena kita tidak mengetahui berapa nilai *x\*,* x dipilih secara acak, (contohnya x adalah 0).

#### **Cara menentukan nilai awal x Newton**

##### **Alasan penentuan**

Tidak ada cara optimal untuk menentukan nilai awal x, sehingga harus ditebak secara *random*. Tetapi bila tebakan nilai x tersebut dengan nilai akar sesungguhnya, maka iterasi tidak akan banyak. Dan lebih baik jika tebakan nilai x bukan merupakan nilai pada asimtot horizontal ataupun lokal maxima, karena perhitungan pada iterasi bisa meleset sehingga divergen.

#### **Cara menentukan nilai awal x Modified Newton**

##### **Alasan penentuan**

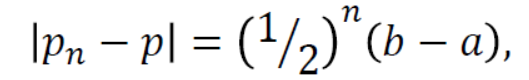
Untuk bisa convergen sebenarnya x harus mendekat x\*, karena tidak dapat mengetahui x yang mendekati x\* maka penentuannya tidak lain adalah secara random.

# Analisis

### **Bisection Method**

#### **Kaitan algoritma dengan teori**

Algoritma bisection method akan mencari akar dari f(x) yang merupakani persamaan non-linear. Misalnya terdapat fungsi f(x) yang kontinyu pada interval [a,b], dimana f(a) dan f(b) memiliki tanda yang berbeda (f(a) ∙ f(b) < 0) . Menurut *Intermediate Value Theorem* (IVT), ada sebuah p pada (a,b) dengan f(p) = 0. Bagi dua interval [a,b] tersebut dan terapkan IVT berulang kali hingga melanggar *stopping criteria* ( |(b-a)| < toleransi atau |f(p)| < toleransi). Bisection method akan men-generate barisan {pn}∞n= 1 mendekati 0 p pada f(x) dengan



Ketika n

#### **Kelebihan algoritma**

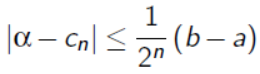
1. Bisection method menjamin konvergensi, jika interval [a,b] yang diberikan tepat
2. Error bound akan menurun hingga setengahnya pada setiap iterasi

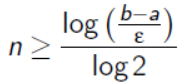
#### **Kekurangan algoritma**

1. Lambat untuk mencapai konvergen
2. Tidak dapat mendeteksi banyak akar

#### **Akurasi**

adalah nilai dari suatu akar, a ≤ ≤ b. an, bn and cn adalah nilai dari a, b dan c iterasi ke- n dari algoritma. *Error bound* dari cn dinyatakan sebagai berikut:

Pertidaksamaan ini dapat memberikan jumlah iterasi yang diperlukan untuk akurasi 

Akurasi dipengaruhi oleh toleransi yang diberikan, semakin besar toleransi maka semakin banyak iterasi yang dibutuhkan untuk mendapatkan akar, tetapi semakin teliti hasil yang didapatkan. Selain itu, nilai interval [a,b] juga mempengaruhi banyaknya iterasi dalam mendapatkan akar serta akurasi dari hasil akar yang diperoleh.

### 

### **Fixed Point Method**

#### **Kaitan algoritma dengan teori**

Algoritma fixed point akan mengiterasi fungsi g(x) dengan tebakan awal x0. Iterasi tersebut berbentuk *xk+1 = g(xk).*

#### **Kelebihan algoritma**

Mudah untuk diimplementasikan.

#### **Kekurangan algoritma**

Sulit untuk mencari g(x) yang memenuhi syarat konvergen

#### **Akurasi**

Akurasi metode ini ditentukan oleh TOL (toleransi) yang diberikan ke fungsi fixed\_point. Semakin kecil TOL, semakin kecil perbedaan xk+1 dan xk yang harus dicapai.

### **Newton Method**

#### **Kaitan algoritma dengan teori**

Newton method adalah salah satu cara untuk mendapatkan akar persamaan non-linear. Dengan memanfaatkan persamaan linear berupa garis singgung di sebuah titik x, maka persamaan non-linear tersebut bisa didekatkan dengan persamaan linear. Dengan melakukan iterasi, maka dari tebakan awal x, bisa didapatkan nilai akar sesungguhnya.

#### **Kelebihan algoritma**

* Kecepatan konvergen relatif cepat
* Tebakan nilai awal x yang dekat dengan akar/x\* akan membuat komputasi cepat
* Mudah diimplementasi

#### **Kekurangan algoritma**

* Harus menebak nilai awal x secara random (tidak optimal)
* Harus mencari turunan pertama dari persamaan nonlinear terlebih dahulu
* Pemilihan tebakan awal x yang salah dapat membuat tidak menemukan akar/x\* (divergen)
* Konvergensi lambat untuk akar yang multiplisitas lebih dari 1

#### **Akurasi**

Akurasi dipengaruhi oleh toleransi yang diberikan, semakin kecil toleransi maka semakin besar iterasi yang dibutuhkan untuk mendapatkan akar, tetapi semakin teliti hasil yang didapatkan.

### **Modified Newton Method**

#### **Kaitan algoritma dengan teori**

Metode Newton termodifikasi adalah suatu metode pencarian akar yang didasarkan pada prinsip iterasi metode Newton standar , yaitu pendekatan fungsi linear. Skema Newton termodifikasi diperoleh dengan mempertinggi tingkat keakuratan metode Newton standar dengan memperhatikan fungsi tak linear yang akan ditentukan akarnya, salah satunya dengan memanfaatkan penggunaan fungsi turunan pertama dan kedua. Untuk secara detailnya mengapa penerapan tersebut dapat berpengaruh secara signifikan, saya pribadi belum melakukan pencarian yang lebih jauh. Dari hasil percobaan tersebut terlihat (jika dibandingkan dengan hasil eksperimen menggunakan metode newton biasa) kecepatan konvergensinya jauh lebih cepat dibanding dengan metode newton biasa. Dalam percobaan ini, ukuran waktu sulit ditentukan sehingga digunakan jumlah iterasi sebagai satuan pengukur waktu, karena rata-rata waktu yang dibutuhkan untuk menjalankan algoritma ini kurang dari 0 detik.

#### **Kelebihan algoritma**

* Mempertahankan kecepatan konvergensi pada kasus terdapat akar kembar, pada metode newton biasa, adanya akar kembar kecepatan konvergensi lebih lambat.

#### **Kekurangan algoritma**

* Perlu mencari turunan pertama dan turunan kedua fungsi

#### **Akurasi**

* Akurasi yang digunakan adalah seberapa besar perbedaan persamaan jika dimasukkan dengan nilai x (yaitu f(xi)) yang ditemukan pada setiap iterasi dibandingkan dengan solusi yang diharapkan (yaitu f(x) = 0).

# Kesimpulan

Dalam permasalahan pada sistem persamaan non-linear, solusi tidak selalu dapat ditemukan. Pada eksperimen sebelumnya telah diterapkan beberapa algoritma untuk menyelesaikan permasalahan non-linear dimana masing-masing algoritma memiliki keterbatasan dan kelebihan yang berbeda-beda. Dari hasil eksperimen yang telah dilakukan pendekatan modified newton memiliki tingkat konvergensi yang lebih optimal dibandingkan dengan algoritma yang lainnya.

# 

# Kontribusi

# 

|  |  |  |  |
| --- | --- | --- | --- |
| No | Nama | Tugas | Persentase(%) |
| 1 | Sarah Dewi Fadlia | Implementasi Bisection Method | 22% |
| 2 | Gilang Gumilar | Implementasi Fixed point Method | 22% |
| 3 | Stefanus James | Implementasi Newton’s Method | 22% |
| 4 | Gunawan | Implementasi Modified Newton Method, Mengerjakan soal nomor 2, Membuat helper method untuk keempat algoritma. | 35% |

# 

# Daftar Pustaka

[1] T. Basaruddin., Komputasi Numerik, Depok: Lembaga Penerbit Fakultas Ekonomi Universitas Indonesia, 2007, pp. 74-76

[2] Zhiliang Xu, "2.1 The Bisection Method", nd.edu, Available: https://www3.nd.edu/~zxu2/acms40390F13/Lec-2.1.pdf. [Accessed: Mar. 22, 2018].

[3] Dr Dana Mackey, "Numerical Methods", maths.dit.ie, Oct. 3, 2014. [online]. Available: http://www.maths.dit.ie/~dmackey/lectures/Roots.pdf. [Accessed: Mar. 23, 2018].

[4]

[**http://mathworld.wolfram.com/NewtonsMethod.html**](http://mathworld.wolfram.com/NewtonsMethod.html)

[**http://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx**](http://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx)

[**http://www.ugrad.math.ubc.ca/coursedoc/math100/notes/approx/newton.html**](http://www.ugrad.math.ubc.ca/coursedoc/math100/notes/approx/newton.html)

[**http://macs.citadel.edu/chenm/343.dir/11.dir/lect2\_4.pdf**](http://macs.citadel.edu/chenm/343.dir/11.dir/lect2_4.pdf)[**https://m.njit.edu/Undergraduate/Matlab/M111MATLAB2S08/**](https://m.njit.edu/Undergraduate/Matlab/M111MATLAB2S08/)