

PERSAMAAN NON LINEAR

Mengenai:

1. Metode Bagi Dua (Bisection)
2. Metode Regulafalsi
3. Metode Bagi Dua (Bisection) menggunakan Phyton
4. Metode Regulafalsi menggunakan Phyton

Pendahuluan

Persamaan non linear banyak digunakan dalam bidang teknik maupun sains. Secara umum, semua persamaan pada permasalahan dalam bentuk ini akan diubah menjadi bentuk: $f(x) = 0$, dengan f yang merupakan bentuk fungsi non linear dari variabel x , contoh persamaannya adalah bentuk persamaan Beattie-Bridgeman yang merupakan persamaan gas riil sebagai hubungan antara suhu-tekanan-volume, dengan bentuk:

$$PV = RT + \frac{\beta}{V} + \frac{\gamma}{V^2} + \frac{\delta}{V^3}$$

Metode penyelesaian akar-akar persamaan secara komputasi numerik dapat dibedakan menjadi dua macam, yaitu metode pengurungan (bracketing method) dan metode terbuka (open method). Metode pengurungan merupakan metode yang memerlukan dua titik sebagai tebakan awal dengan akar-akar persamaan yang diperkirakan berada di antara kedua titik tebakan awal tersebut. Di sisi lain, metode lain yang lebih sederhana adalah metode grafis. Metode ini akan menggambarkan hubungan $f(x)$ dan x dari fungsi tersebut memotong sumbu x . khususnya untuk permasalahan dengan jumlah variable yang banyak.

Metode pengurungan secara garis besar bekerja seperti berikut: dimulai dengan suatu interval $[a,b]$ dimana $f(a).f(b) < 0$, lalu diperkecil interval secara bertahap sampai akurasi yang diinginkan didapatkan. Metode pengurungan ini selalu berhasil mendapatkan nilai estimasi dari akar persamaan. Pada modul ini dibahas mengenai jenis-jenis metode pengurungan yaitu

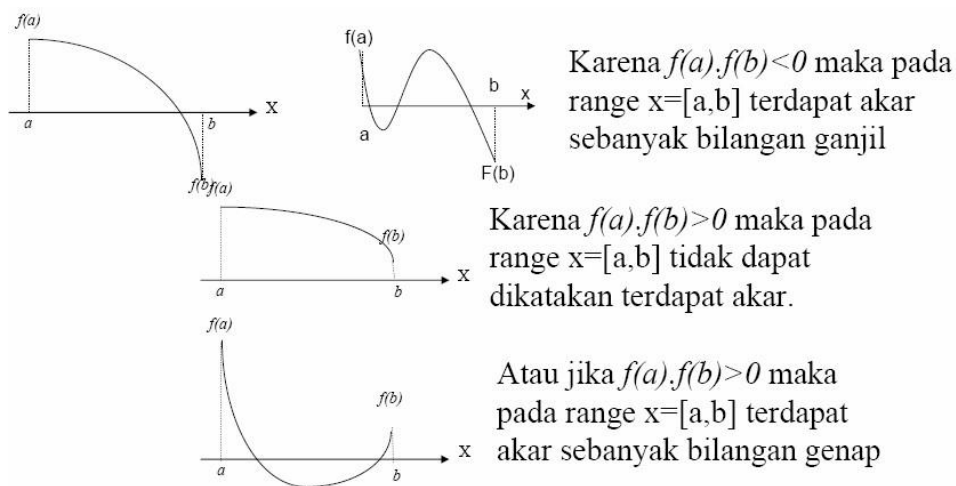
metode bagi dua (bisection) dan metode regulafalsi.

Beberapa contoh permasalahan yang memerlukan penyelesaian persamaan non linier sebagai kuncinya adalah sebagai berikut:

- Penentuan nilai maksimal dan minimal fungsi non linier.
- Perhitungan nilai konstanta pada matrik dan determinan, yang biasanya muncul dalam permasalahan sistem linier, Bisa digunakan untuk menghitung nilai eigen.
- Penentuan titik potong beberapa fungsi non linier, yang banyak digunakan untuk keperluan perhitungan-perhitungan secara grafis.
- Penyelesaian persamaan non linier adalah penentuan akar – akar persamaan non linier.
- Akar sebuah persamaan $f(x) = 0$ adalah nilai – nilai x yang menyebabkan nilai $f(x)$ sama dengan nol.
- Akar persamaan $f(x)$ adalah titik potong antara kurva $f(x)$ dan sumbu X .

Teorema Penyelesaian Persamaan Non Linier

Suatu range $x=[a,b]$ mempunyai akar bila $f(a)$ dan $f(b)$ berlawanan tanda atau memenuhi $f(a).f(b)<0$.



Gambar 2.2. Teorema Penyelesaian Persamaan Non Linier

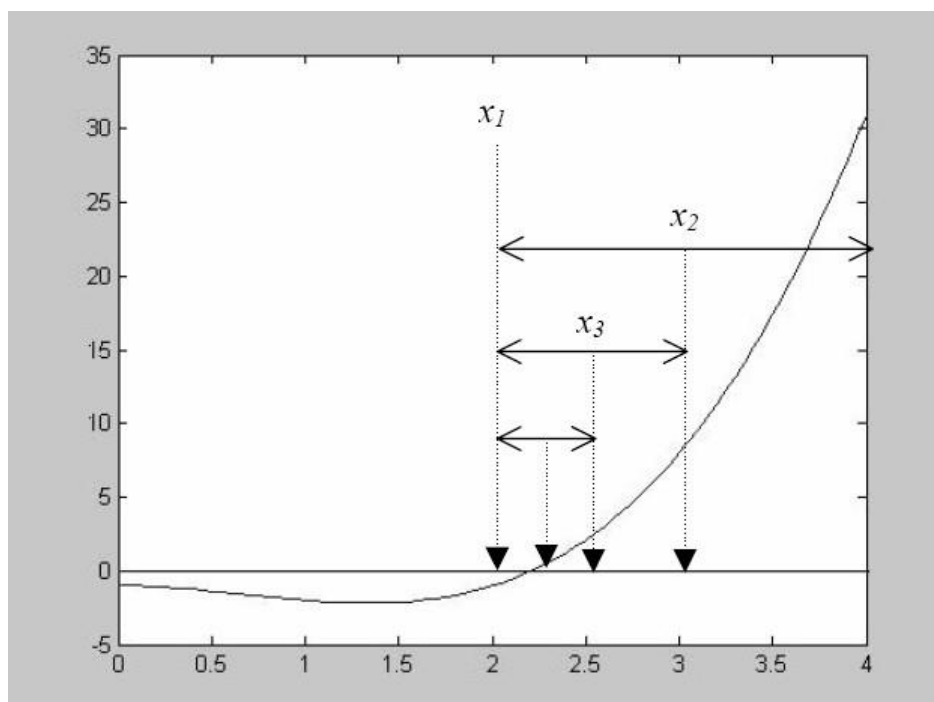
2.1 Metode Bagi Dua (*Bisection*)

Metode bagi dua (*bisection*) merupakan metode yang paling sederhana. Metode ini diawali dengan menebak dua nilai, yaitu x_a dan x_b , dengan beranggapan bahwa nilai akar yang dicari berada di antara keduanya. Oleh karena itu, metode ini disebut metode pengurangan.

- Metode biseksi ini membagi range menjadi 2 bagian, dari dua bagian ini dipilih mana yang mengandung dan bagian yang tidak mengandung akar dibuang. Hal ini dilakukan berulang-ulang hingga diperoleh akar persamaan.
- Untuk menggunakan metode biseksi, tentukan batas bawah (a) dan batas atas (b). Kemudian dihitung nilai tengah : $x = \frac{(a+b)}{2}$
- Dari nilai x ini perlu dilakukan pengecekan keberadaan akar : $f(x)$. $f(a) < 0$, maka $b = x$, $f(b) = f(x)$, a tetap

$f(x) \cdot f(a) > 0$, maka $a = x$, $f(a) = f(x)$, b tetap

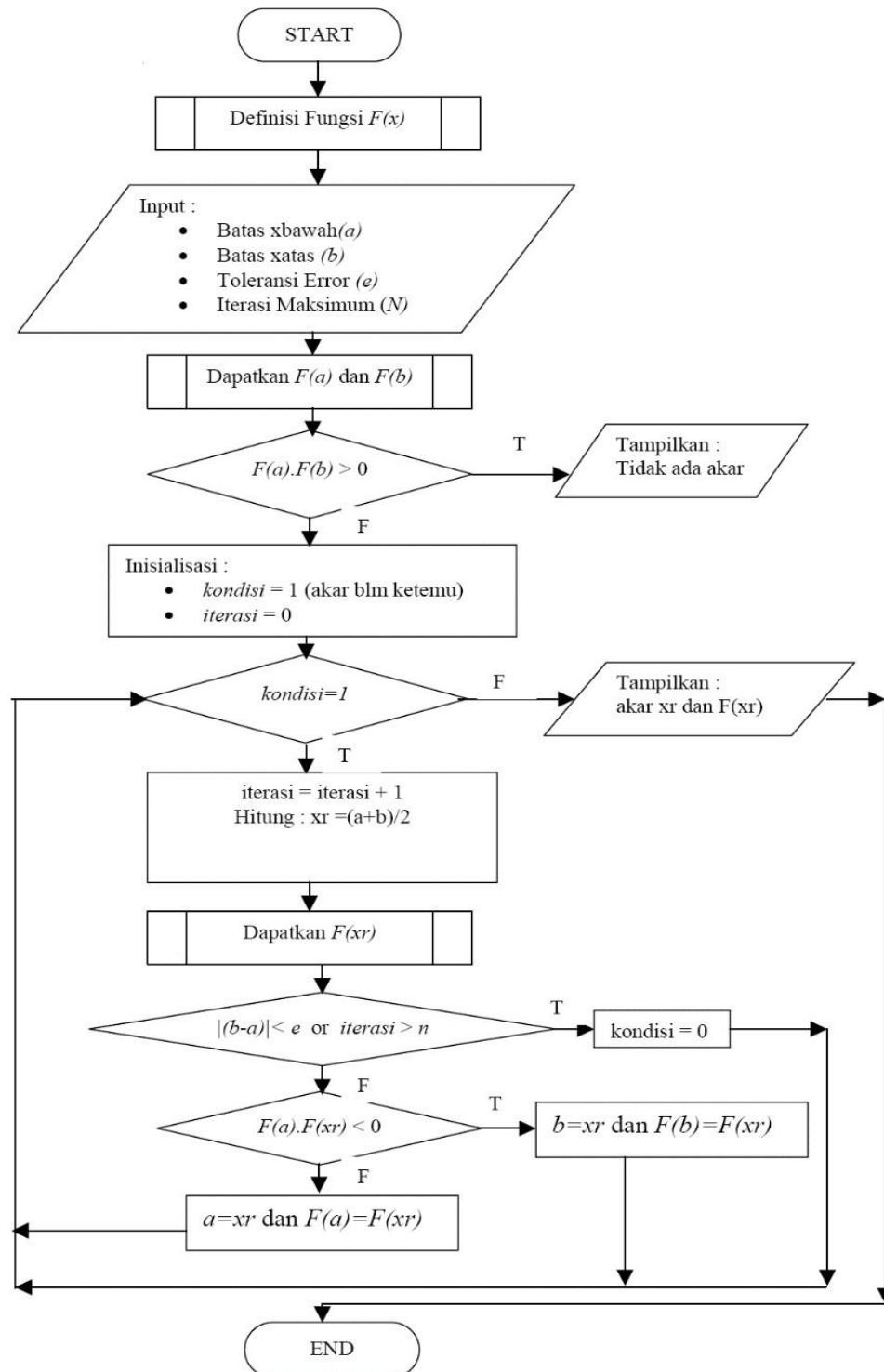
- Setelah diketahui dibagian mana terdapat akar, maka batas bawah & batas atas diperbarui sesuai dengan range dari bagian yang mempunyai akar.



Gambar 2.3. Grafik Metode Bisection

Algoritma Metode Biseksi :

1. Definisikan fungsi $f(x)$ yang akan dicari akarnya
2. Tentukan nilai a dan b
3. Tentukan toleransi e dan iterasi maksimum N
4. Hitung $f(a)$ dan $f(b)$
5. Jika $f(a).f(b) > 0$ maka proses dihentikan karena tidak ada akar, bila tidak dilanjutkan
6. Hitung $x = \frac{a+b}{2}$ Hitung $f(x)$
7. Bila $f(x).f(a) < 0$ maka $b = x$ dan $f(b) = f(x)$, bila tidak $a = x$ dan $f(a) = f(x)$
8. Jika $|b - a| < e$ atau iterasi $>$ iterasi maksimum maka proses dihentikan dan didapatkan akar $= x$, dan bila tidak, ulangi langkah 6.



Gambar 2.4. Flowchart Metode Biseksi

Contoh 2.1:

Diketahui suatu fungsi $f(x) = e^x + 2x$ dengan nilai awal $[-1,0]$ dan toleransi 0,05 cari akar akar persamaan non-linear tersebut!

Jawab:

1. $f(x) = e^x + 2x$

$$f(-1) = e^{-1} + 2(-1)$$

$$f(0) = e^0 + 2(0)$$

$$f(-1) = -1,6321206$$

$$f(0) = 1$$

2. Dapat dilihat selang $[-1,0]$ memenuhi syarat $f(a) \cdot f(b) < 0$ maka selang tersebut dapat mulai digunakan

3. Mulai lakukan perhitungan

Iterasi 1

(i) Cari nilai x

$$x = \frac{a + b}{2} = \frac{-1 + 0}{2} = -0.5$$

(ii) Cari nilai f (x)

$$f(x) = e^x + 2(x)$$

$$f(-0.5) = e^{-0.5} + 2(-0.5) = -0.3934693$$

(iii) Cari nilai f(a).f(x)

$$f(a) \cdot f(x) = f(-1) \cdot f(-0.5)$$

$$f(-1) \cdot f(-0.5) = (-1.6321206) \cdot (-0.3934693)$$

$$f(-1) \cdot f(-0.5) = 0.6421893$$

(iv) Tentukan selang berikutnya

$$f(a) \cdot f(x) > 0 \text{ maka selang selanjutnya adalah } [x, b]$$

$$a = x ; b = b$$

$$a = -0.5; b = 0$$

(v) Cek apakah sudah memenuhi toleransi

$$|a - b| = |-0.5 - 0| = 0.5$$

apakah $|a - b| < \text{toleransi}$?

Tidak, lanjut melakukan Iterasi 2 dengan selang baru

- **Iterasi 2**

(i) Cari nilai x

$$x = \frac{a+b}{2} = \frac{-0.5+0}{2} = -0.25$$

(ii) Cari nilai f (x)

$$f(x) = e^x + 2(x)$$

$$f(-0.25) = e^{-0.25} + 2(-0.25) = 0.27880$$

(iii) Cari nilai f(a).f(x)

$$f(a) \cdot f(x) = f(-0.5) \cdot f(-0.25)$$

$$f(-0.5) \cdot f(-0.25) = (-0.3934693) \cdot (0.27880)$$

$$f(-0.5) \cdot f(-0.25) = -0.1096996$$

(iv) Tentukan selang berikutnya

$$f(a) \cdot f(x) < 0 \text{ maka selang selanjutnya adalah } [a, x]$$

$$a = a ; b = x$$

$$a = -0.5; b = -0.25$$

- (v) Cek apakah sudah memenuhi toleransi

$$|a - b| = |-0.5 - (-0.25)| = 0.25$$

apakah $|a - b| < \text{toleransi}$?

Tidak, lanjut melakukan Iterasi 3 dengan selang baru

- **Iterasi 3**

- (i) Cari nilai x

$$x = \frac{a+b}{2} = \frac{-0.5+(-0.25)}{2} = -0.375$$

- (ii) Cari nilai f (x)

$$f(x) = e^x + 2(x)$$

$$f(-0.375) = e^{-0.375} + 2(-0.375) = -0.06271$$

- (iii) Cari nilai f(a).f(x)

$$f(a) \cdot f(x) = f(-0.5) \cdot f(-0.375)$$

$$f(-0.5) \cdot f(-0.375) = (-0.3934693) \cdot (-0.06271)$$

$$f(-0.5) \cdot f(-0.375) = 0.0246745$$

- (iv) Tentukan selang berikutnya

$$f(a) \cdot f(x) > 0 \text{ maka selang selanjutnya adalah } [x, b]$$

$$a = x ; b = b$$

$$a = -0.375; b = -0.25$$

- (v) Cek apakah sudah memenuhi toleransi

$$|a - b| = |-0.375 - (-0.25)| = 0.125$$

apakah $|a - b| < \text{toleransi}$?

Tidak, lanjut melakukan Iterasi 4 dengan selang baru

- **Iterasi 4**

(i) Cari nilai x

$$x = \frac{a + b}{2} = \frac{-0.375 + (-0.25)}{2} = -0,3125$$

(ii) Cari nilai f (x)

$$f(x) = e^x + 2(x)$$

$$f(-0.3125) = e^{-0.3125} + 2(-0.3125) = 0.10662$$

(iii) Cari nilai f(a).f(x)

$$f(a) \cdot f(x) = f(-0.375) \cdot f(-0.3125)$$

$$f(-0.375) \cdot f(-0.3125) = (-0.06271) \cdot (0.10662)$$

$$f(-0.375) \cdot f(-0.3125) = -0.0066861$$

(iv) Tentukan selang berikutnya

$$f(a) \cdot f(x) < 0 \text{ maka selang selanjutnya adalah } [a, x]$$

$$a = a ; b = x$$

$$a = -0.375; b = -0.3125$$

(v) Cek apakah sudah memenuhi toleransi

$$|a - b| = |-0.375 - (-0.3125)| = 0.0625$$

$$\text{apakah } |a - b| < \text{toleransi} ?$$

Tidak, lanjut melakukan Iterasi 5 dengan selang baru

- **Iterasi 5**

(i) Cari nilai x

$$x = \frac{a + b}{2} = \frac{-0.375 + (-0.3125)}{2} = -0,34375$$

(ii) Cari nilai f (x)

$$f(x) = e^x + 2(x)$$

$$f(-0.34375) = e^{-0.34375} + 2(-0.34375) = 0.02161$$

(iii) Cari nilai f(a).f(x)

$$f(a) \cdot f(x) = f(-1) \cdot f(-0.5)$$

$$f(-0.375) \cdot f(-0.34375) = (-0.06271) \cdot (0.02161)$$

$$f(-0.375) \cdot f(-0.34375) = -0.0013552$$

(iv) Tentukan selang berikutnya

$$f(a) \cdot f(x) < 0 \text{ maka selang selanjutnya adalah } [x, b]$$

$$a = a ; b = x$$

$$a = -0.375; b = -0,34375$$

(v) Cek apakah sudah memenuhi toleransi

$$|a - b| = |-0.375 - (-0.34375)| = 0.03125$$

$$\text{apakah } |a - b| < \text{toleransi} ?$$

Ya, Iterasi berhenti. Nilai hampiran akar adalah -0.34375

Keuntungan Biseksi

Selalu berhasil menemukan akar (solusi) yang dicari, atau dengan kata lain selalu konvergen.

Kelemahan Biseksi

- Bekerja sangat lambat. Tidak memandang bahwa sebenarnya akar atau solusi yang dicari telah berada dekat sekali dengan x_0 ataupun x_1 .
- Metode biseksi hanya dapat dilakukan apabila ada akar persamaan pada interval yang

diberikan.

- Jika ada beberapa akar pada interval yang diberikan maka hanya satu akar saja yang dapat ditemukan.
- Memiliki proses iterasi yang banyak sehingga memperlama proses penyelesaiannya.

2.2 Metode RegulaFalsi

Metode posisi palsu (false position), yang dalam bahasa Latin disebut regula falsi, merupakan perbaikan bagi metode bagi dua (bisection). Adapun pengertian metode regula falsi yaitu metode pencarian akar persamaan dengan memanfaatkan kemiringan dan selisih tinggi dari dua titik batas range. Pada metode bagi dua, nilai x_c diperkirakan dengan membagi interval diantara kedua x_b dan x_a tanpa melihat besarnya nilai $f(x_b)$ dan $f(x_a)$.

Hal tersebut akan menyebabkan proses iterasi menjadi relatif banyak. Berbeda dengan metode bisection, metode regula falsi akan menarik garis lurus pada kedua interval sehingga dapat memperbaiki perkiraan nilai x_c . Oleh karena itu, metode ini juga disebut metode interpolasi linear.

Metode ini bekerja secara iterasi dengan melakukan update range. Titik pendekatan yang dipakai adalah

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

Prinsip dari metode ini didasarkan pada interpolasi linier. Perbedaannya dengan metode bagi dua terletak pada pencarian akar persamaan setelah akar tersebut dikurung oleh dua harga taksiran awal. Selanjutnya dilakukan interpolasi linier pada ujung-ujung titik untuk memperoleh pendekatan harga akar. Jadi, jika fungsi tersebut dapat didekati dengan cara interpolasi linier, maka akar-akar taksiran tersebut memiliki ketelitian yang tinggi, akibatnya iterasi dapat mencapai konvergensi ke arah harga akar pendekatan dengan cepat.

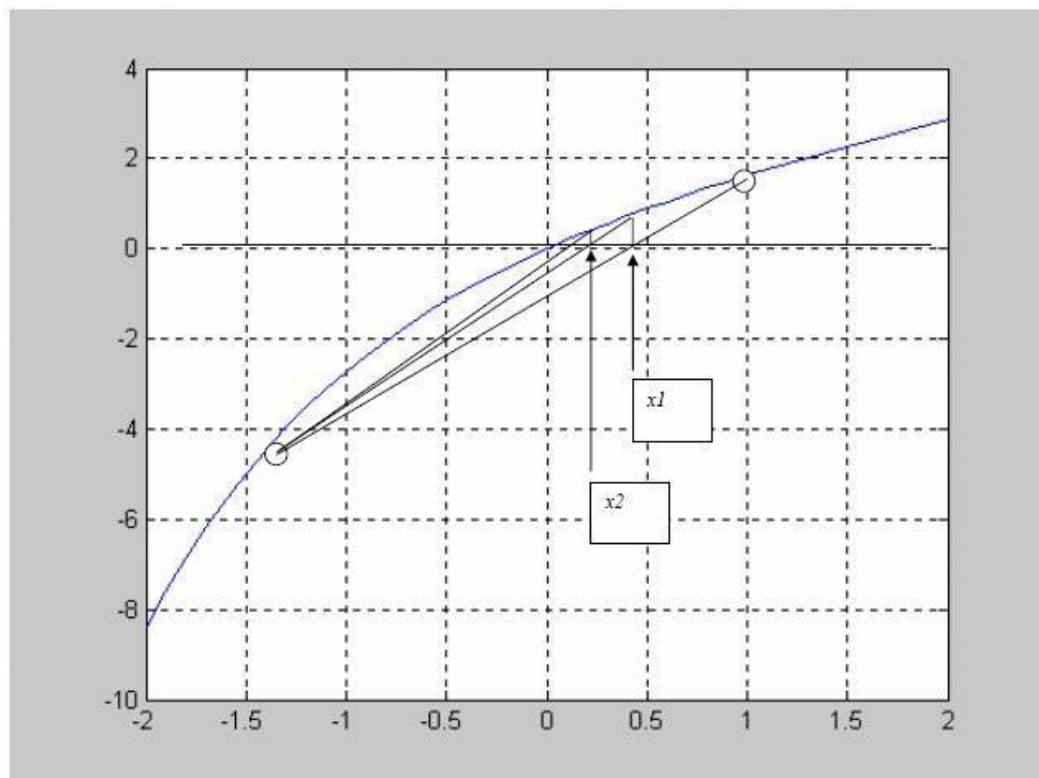
Penetapan interval baru:

bila $f(x) \times f(a) < 0$ maka intervalnya menjadi $[a, x]$

bila $f(x) \times f(a) > 0$ maka intervalnya menjadi $[x, b]$

Pengulangan/iterasi mencari x_2 dan interval baru dilakukan berdasarkan nilai toleransi atau bila akarnya belum ditemukan. Sebaiknya nilai toleransi secara relatif mengacu pada : error

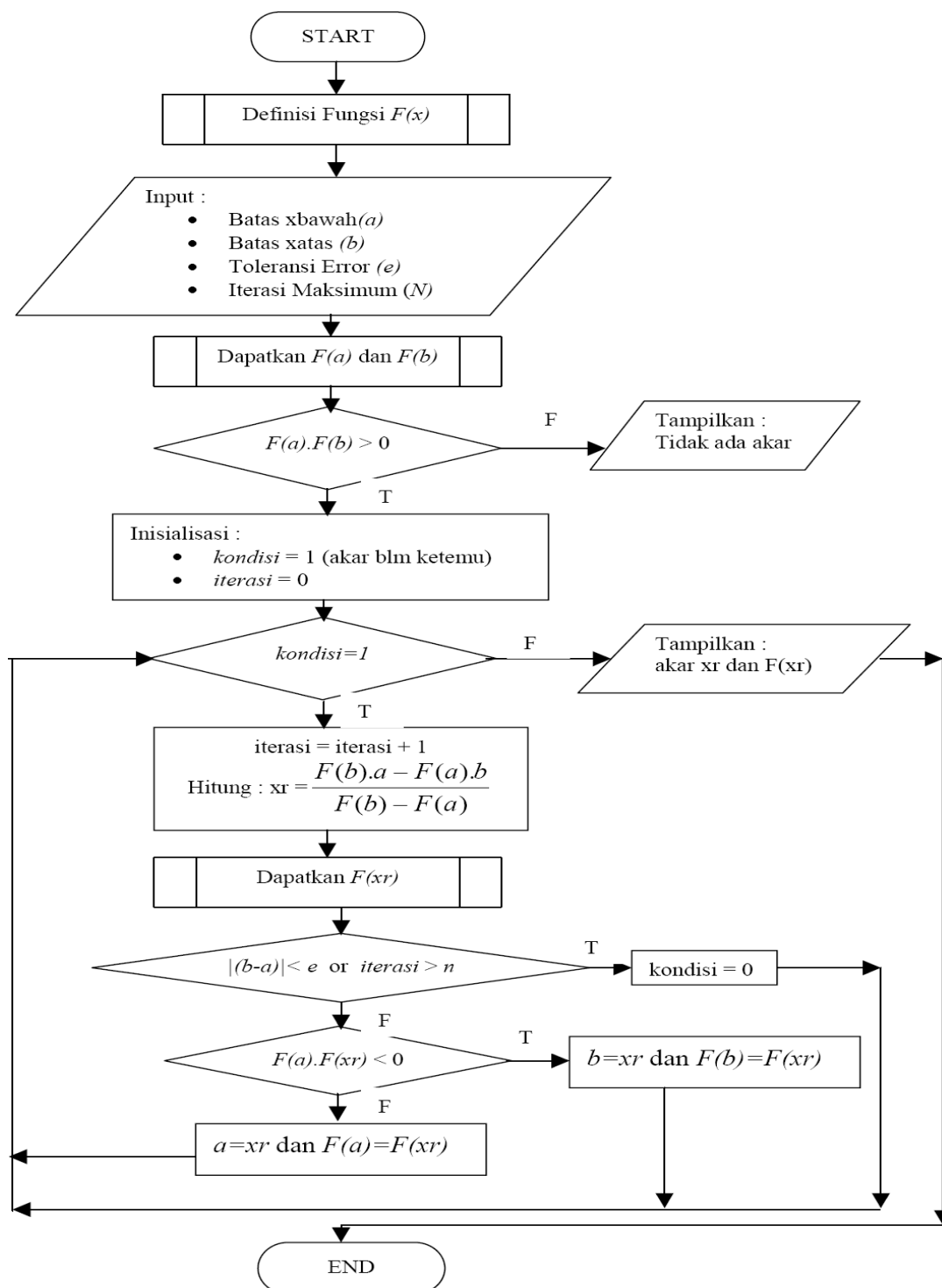
aproksimasi



Gambar 2.5 Error aproksimasi

Algoritma Metode Regula-falsi :

1. Definiskan fungsi $f(x)$
2. Tentukan rentang untuk x yang berupa batas bawah (a) dan batas atas (b)
3. Tentukan toleransi (e) dan iterasi maksimum (n)
4. Hitung $f(a)$ = fungsi(a) dan $f(b)$ = fungsi(b)
5. Untuk iterasi $i = 1$ s/d n atau error $> e$
6. Hitung nilai x dengan persamaan $x = \frac{f(b).a - f(a).b}{f(b) - f(a)}$ dan Hitung f_x = fungsi(x)
7. Hitung error = $|f_x|$
8. Jika $f(x).f(a) < 0$ maka $b = x$ dan $f(b) = f(x)$, jika tidak $a = x$ dan $f(a) = f(x)$.
9. Akar persamaan adalah x



Gambar 2.6. Flowchart Metode Regula-falsi

Contoh 2.2:

Diketahui suatu fungsi $f(x) = e^x + 2x$ dengan nilai awal $[-1, 0]$ dan toleransi 0,05 cari akar akarpersamaan non-linear tersebut!

Jawab:

- **Iterasi 1**

(i) Hitung $f(a)$ dan $f(b)$

$$f(x) = e^x + 2x$$

$$f(-1) = e^{-1} + 2(-1)$$

$$f(-1) = -1,6321206$$

$$f(0) = e^0 + 2(0)$$

$$f(0) = 1$$

(ii) Cari nilai x

$$x = \frac{f(b) \times a - f(a) \times b}{f(b) - f(a)} = \frac{(1 \times -1) - (-1,6321206 \times 0)}{1 - (-1,6321206)} = -0.3799$$

(iii) Cari nilai $f(x)$

$$f(x) = e^x + 2(x)$$

$$f(-0.3799) = e^{-0.3799} + 2(-0.3799) = -0.0759$$

(iv) Hitung toleransi

$$\text{Toleransi} = |f(x)|$$

$$\text{Toleransi} = |-0.0759| = 0.0759$$

(v) Cari nilai $f(x) \cdot f(a)$ untuk menentukan interval akar

$$f(x) \cdot f(a) = f(-0.3799) \cdot f(-1)$$

$$f(-0.3799) \cdot f(-1) = (-0.0759) \cdot (-1.6324206)$$

$$f(-0.3799) \cdot f(-1) = 0.12390072354$$

Karena $f(x) \cdot f(a) > 0$ maka selang selanjutnya adalah $[x, b]$

$$a = x; b = b$$

$$a = -0.3799; b = 0$$

- **Iterasi 2**

Tentukan rentang untuk x yang berupa batas bawah a dan batas atas b yang didapat dari iterasi sebelumnya dan hitung f(a) dan f(b)

$$a = -0.3799; b = 0$$

$$f(x) = e^x + 2x$$

$$f(-0.3799) = e^{-0.3799} + 2(-0.3799) \qquad f(0) = e^0 + 2(0)$$

$$f(-1) = -0.0759 \qquad f(0) = 1$$

(i) Hitung nilai x

$$x = \frac{f(b) \times a - f(a) \times b}{f(b) - f(a)} = \frac{(1 \times -0.3799) - (-0.0759 \times 0)}{1 - (-0.0759)} = -0.3531$$

(ii) Cari nilai f (x)

$$f(x) = e^x + 2(x)$$

$$f(-0.3531) = e^{-0.3531} + 2(-0.3531) = -0.0037$$

(vi) Hitung toleransi

$$\text{Toleransi} = |f(x)|$$

$$\text{Toleransi} = |-0.0037| = 0.0037$$

(vii) Cari nilai f(x).f(a) untuk menentukan interval akar

$$f(x) \cdot f(a) = f(-0.3531) \cdot f(-0.3799)$$

$$f(-0.3531) \cdot f(-0.3799) = (-0.0037) \cdot (-0.0759)$$

$$f(-0.3531) \cdot f(-0.3799) = 0.00028083$$

Karena $f(x) \cdot f(a) \leq 0$ maka hasil akarnya adalah -0.3531

KEUNTUNGAN REGULAFALSI

Selalu berhasil menemukan akar (solusi) yang dicari, atau dengan kata lain selalu konvergen.

KELEMAHAN REGULAFALSI

- Hanya salah satu titik ujung interval (x_0 atau x_1) yang bergerak menuju akar dan yang lainnya selalu tetap untuk setiap iterasi.
- Sehingga mungkin $[x_0, x_1]$ masih cukup besar jaraknya bila menggunakan batas

$$|x_1 - x_0| \leq T \text{ padahal } x_0 \rightarrow x_2 \text{ atau } x_1 \rightarrow x_2$$

- Hal tersebut dikenal dengan pendekatan error mutlak. Diperbaiki dengan pendekatan Error relatif :

$$\left| \frac{x_1 - x_2}{x_1} \right| \leq T \quad \text{atau} \quad \left| \frac{x_0 - x_2}{x_0} \right| \leq T$$

2.3 Metode Bagi Dua menggunakan Python

Kita juga dapat menyelesaikan permasalahan sistem persamaan non-linear dengan menerapkan metode bagi dua kedalam kode Python. Untuk implementasi metode bagi dua pada permasalahan contoh 2.1 di atas kita dapat amati pada listing program 2.1 berikut ini.


```

# Import library numpy dan memberikan alias np
import numpy as np

print("Hasil Metode Bisection :")

# Membuat fungsi dengan nama fungsi dan parameter x
def fungsi(x):
    return np.exp(x)+2*x

# Membuat fungsi dengan nama metode_bisection dengan parameter
a, b, toleransi
def metode_bisection(a, b, toleransi):
    i=0
    if fungsi(a) * fungsi(b) > 0:
        print("Fungsi Tidak Memiliki Akar.")
    else:
        while(abs(b-a)) > toleransi:
            x = (a + b) / 2.0
            i = i + 1
            print('{:2d}          {:3.4f}          {:3.4f}          {:3.4f}'
                  '{:3.4f}  {:3.4f}          {:3.4f}')
            .format(i, a, b, x, fungsi(a),
                  fungsi(b),fungsi(x)))

            if fungsi(x) == 0:
                return(x)
            elif fungsi(a) * fungsi(x) < 0:
                b = x
            else:
                a = x
        return(x)

print(" i      a              b              x              f(a)              f(b)
f(x) ")

# Memanggil fungsi metode_bisection dengan nilai a, b dan
toleransi
hasil = metode_bisection(-1,0,0.05)
print("Hasil akarnya adalah:", hasil)

```

Listing Program 2.1: Listing program metode_bisction.py

```

metode_bisection.py - C:\Users\gundar\metnum\Metode Numerik\metode_bisection.py (3.10.0)
File Edit Format Run Options Window Help
# Impor library numpy dan memberikan alias np
import numpy as np

print("Hasil Metode Bisection :")

# Membuat fungsi dengan nama fungsi dan parameter x
def fungsi(x):
    return np.exp(x)+2*x

# Membuat fungsi dengan nama metode_bisection dengan parameter a, b, toleransi
def metode_bisection(a, b, toleransi):
    i=0
    if fungsi(a) * fungsi(b) > 0:
        print("Fungsi Tidak Memiliki Akar.")
    else:
        while(abs(b-a)) > toleransi:
            x = (a + b) / 2.0
            i = i + 1
            print('{:2d}      {:3.4f}      {:3.4f}      {:3.4f}      {:3.4f}      {:3.4f}'
                  .format(i, a, b, x, fungsi(a), fungsi(b), fungsi(x)))

            if fungsi(x) == 0:
                return(x)
            elif fungsi(a) * fungsi(x) < 0:
                b = x
            else:
                a = x
        return(x)

print(" i      a      b      x      f(a)      f(b)      f(x)")

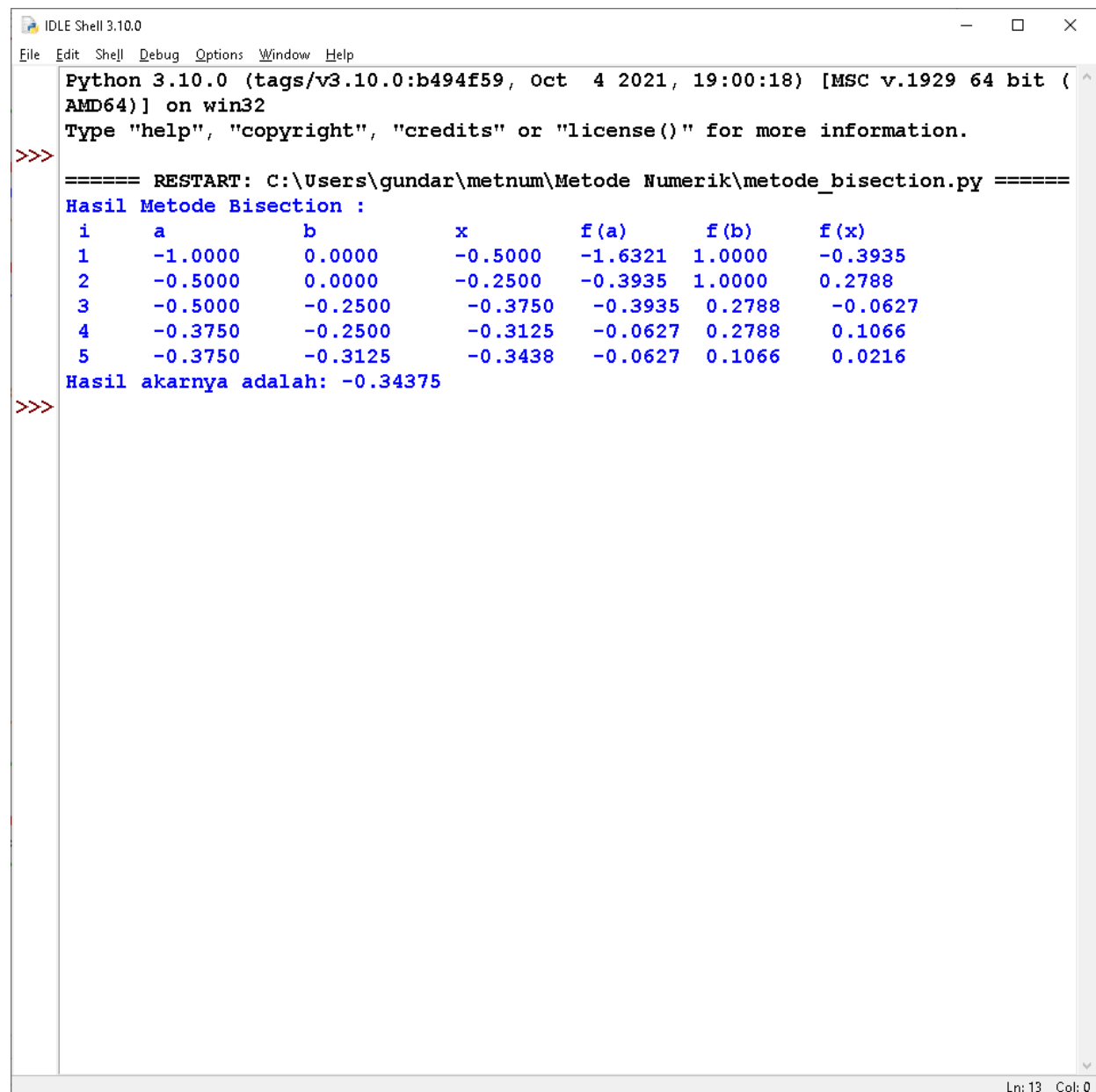
# Memanggil fungsi metode_bisection dengan nilai a, b dan toleransi
hasil = metode_bisection(-1,0,0.05)
print("Hasil akarnya adalah:", hasil)

```

Ln: 35 Col: 0

Gambar 2.7: Listing program metode_bisection.py

Maka output dari program tersebut akan seperti gambar 2.8 dibawah ini



```

Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\gundar\metnum\Metode Numerik\metode_bisection.py =====
Hasil Metode Bisection :
  i      a          b          x          f(a)          f(b)          f(x)
  1     -1.0000      0.0000     -0.5000     -1.6321      1.0000     -0.3935
  2     -0.5000      0.0000     -0.2500     -0.3935      1.0000      0.2788
  3     -0.5000     -0.2500     -0.3750     -0.3935      0.2788     -0.0627
  4     -0.3750     -0.2500     -0.3125     -0.0627      0.2788      0.1066
  5     -0.3750     -0.3125     -0.3438     -0.0627      0.1066      0.0216
Hasil akarnya adalah: -0.34375
>>>

```

Gambar 2.8: Output program metode_bisction.py

2.4 Metode Regula Falsi menggunakan Python

Kita juga dapat menyelesaikan permasalahan sistem persamaan non-linear dengan menerapkan metode regula-falsi kedalam kode Python. Untuk implementasi metode regula-falsi pada permasalahan contoh 2.2 di atas kita dapat amati pada listing program 2.2 berikut ini.

```
# Mengimpor library numpy dan berikan alias sebagai np
import numpy as np

# Membuat fungsi dengan nama fungsi dan berikan variabel parameter
x
def fungsi(x):
    return np.exp(x)+ 2*x

# Membuat fungsi dengan nama regulafalsi dan berikan variabel
parameter a, b, e
def regulafalsi(a, b, e):
    if fungsi(a) * fungsi(b) > 0 :
        print("Inputan anda tidak memenuhi syarat metode numerik
Regulafalsi \n"
              "Silahkan masukan inputan yang benar dimana memenuhi
: "
              "\nfungsi(atas) * fungsi(bawah) < 0")
    x = a
    i = 1
    while abs(fungsi(x))>tol:
        x = (a * fungsi(b) - b * fungsi(a)) / (fungsi(b) -
fungsi(a))
        print('{:2d}   {:3.4f}   {:3.4f}   {:3.4f}   {:3.4f}   {:3.4f}
{:3.4f}'
              .format(i, a, b, x, fungsi(a), fungsi(b),
fungsi(x)))
        if fungsi(x) == 0:
            break
        elif fungsi(x) * fungsi(a) < 0:
            b = x
        else:
            a = x
        i = i + 1
    return x

# Pendefinisian variabel a, b, toleransi
a = -1
b = 0
tol = 0.05

print('i      a      b      x      f(a)      f(b)      f(x) ')
hasil=regulafalsi(a, b, tol)
print("Hasil akarnya adalah: ", round(hasil,4))
```

Listing Program 2.2: listing program metode_regulafalsi.py

```

metode_regulafalsi.py - C:\Users\gundar\metnum\Metode Numerik\metode_regulafalsi.py (3.10.0)
File Edit Format Run Options Window Help
# Mengimpor library numpy dan berikan alias sebagai np
import numpy as np

# Membuat fungsi dengan nama fungsi dan berikan variabel parameter x
def fungsi(x):
    return np.exp(x)+ 2*x

# Membuat fungsi dengan nama regulafalsi dan berikan variabel parameter a, b, e
def regulafalsi(a, b, e):
    if fungsi(a) * fungsi(b) > 0 :
        print("Inputan anda tidak memenuhi syarat metode numerik Regulafalsi \n"
              "Silahkan masukan inputan yang benar dimana memenuhi : "
              "\nfungsi(atas) * fungsi(bawah) < 0")

    x = a
    i = 1

    while abs(fungsi(x))>tol:
        x = (a * fungsi(b) - b * fungsi(a)) / (fungsi(b) - fungsi(a))
        print('{:2d}  {:3.4f}  {:3.4f}  {:3.4f}  {:3.4f}  {:3.4f}  {:3.4f}'
              .format(i, a, b, x, fungsi(a), fungsi(b), fungsi(x)))
        if fungsi(x) == 0:
            break
        elif fungsi(x) * fungsi(a) < 0:
            b = x
        else:
            a = x

        i = i + 1

    return x

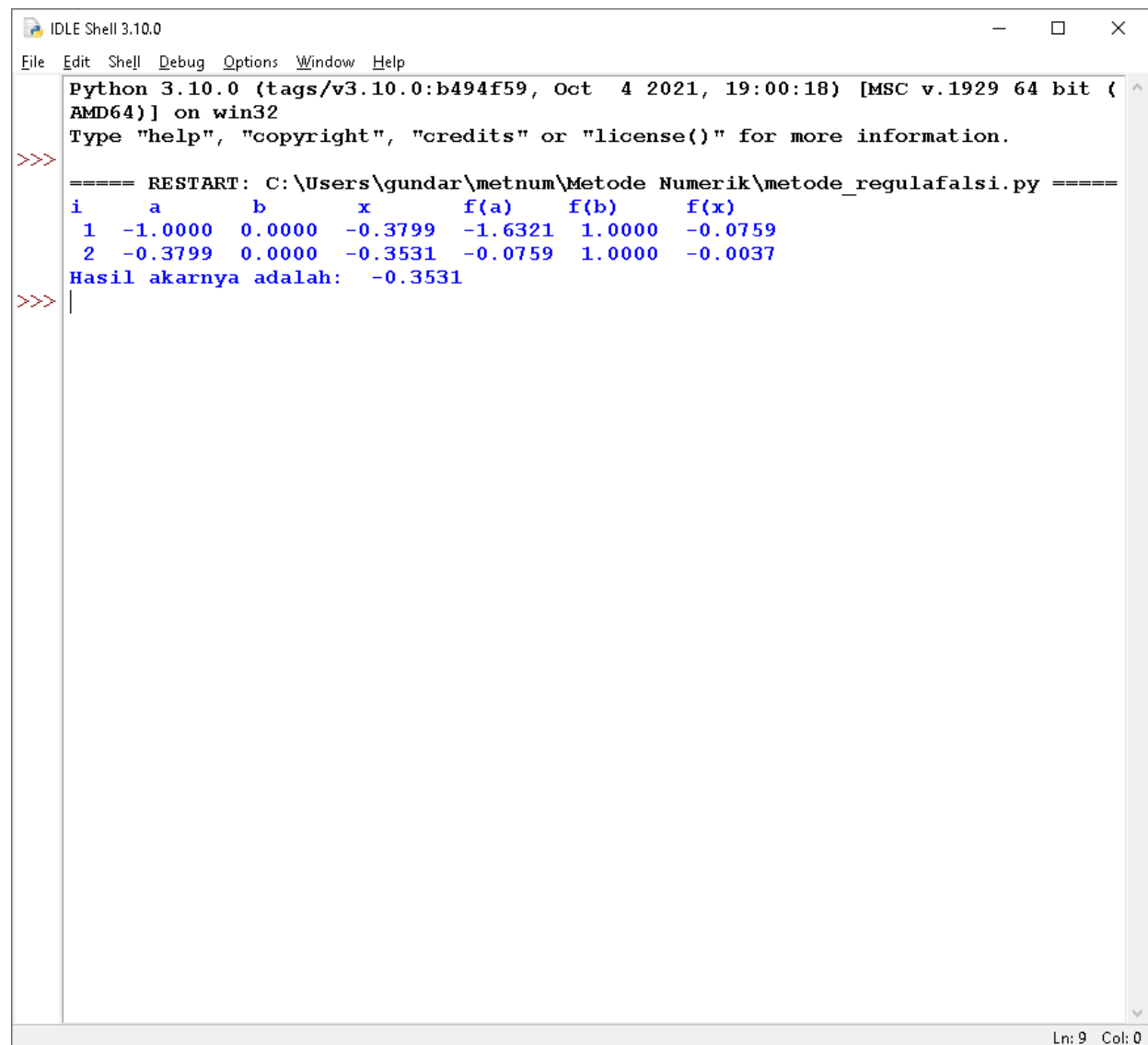
# Pendefinisian variabel a, b, toleransi
a = -1
b = 0
tol = 0.05

print('i      a      b      x      f(a)      f(b)      f(x)')
hasil=regulafalsi(a, b, tol)
print("Hasil akarnya adalah: ", round(hasil,4))

```

Gambar 2.9: Listing program metode_regulafalsi.py

Maka output dari program tersebut akan seperti gambar 2.10 dibawah ini



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\gundar\metnum\Metode Numerik\metode_regulafalsi.py =====
i      a      b      x      f(a)      f(b)      f(x)
1  -1.0000  0.0000 -0.3799 -1.6321  1.0000 -0.0759
2  -0.3799  0.0000 -0.3531 -0.0759  1.0000 -0.0037
Hasil akarnya adalah: -0.3531
>>>
```

Gambar 2.10: Output program metode_regulafalsi.py

Referensi

- Paulus, E. d. (2018). *Perangkat Komputasi Numerik SCILAB berbasis Open-Source: Algoritma dan Penerapannya*. Yogyakarta: Deepublish.
- Rosidi, M. (2019, Desember 23). *Metode Numerik Menggunakan R Untuk Teknik Lingkungan*.
From <https://bookdown.org/>
- Sanjaya WS, M. (2015). *Metode Numerik Berbasis Python*. Yogyakarta: Penerbit Gava Media.
- Sasongko, B. S. (2010). *Metode Numerik dengan Scilab*. Yogyakarta: Penerbit ANDI.
- Sholiun, & Fatomi, Z. S. (2021). *Pemrograman dan Komputasi Numerik*. Yogyakarta: Gadjah Mada University Press.
- Triatmodjo, B. (2006). *Metode Numerik*. Yogyakarta: Betta Offset.