LAPORAN PRAKTIKUM METODE NUMERIK

Penerapan Metode Biseksi dan Regula Falsi dalam Penyelesaian Persamaan Non-Linear

Dosen Pembimbing :
Radhiyatammardhiyyah, S.S.T., M.Sc

Disusuh Oleh:
Rausyanul fikri(2024573010122)

FAKULTAS TEKNOLOGI INFORMASI DAN KOMPUTER
PRODI TEKNIK INFORMATIKA
POLITEKNIK NEGERI LHOKSEUMAWE
2025

Kata Pengantar

Puji syukur kehadirat Allah SWT atas segala rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan praktikum yang berjudul "Penerapan Metode Biseksi dan Metode Regula Falsi untuk Menentukan Akar Persamaan Non-Linear" tepat pada waktunya.

Laporan ini disusun sebagai salah satu tugas pada mata kuliah Metode Numerik yang diampu oleh Ibu Radhiyatammardhiyyah, S.S.T., M.Sc. Tujuan dari praktikum ini adalah untuk memahami cara kerja metode numerik dalam menemukan akar suatu fungsi non-linear menggunakan pendekatan iteratif melalui dua metode klasik, yaitu Biseksi dan Regula Falsi.

Penulis menyadari laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan agar laporan ini menjadi lebih baik di masa mendatang.

Bukit Rata, Oktober 2025

Penulis

Rausyanul fikri

(NIM : 2024573010122)

DAFTAR ISI

Kata Pengantar.....i
DAFTAR ISI.....ii
BAB I PENDAHULUAN.....1
1.1 Latar Belakang.....1
1.2 Tujuan.....1
1.3 Rumusan Masalah.....1
BAB II DASAR TEORI.....2
2.1 Metode Biseksi.....2
2.2 Metode Regula Falsi.....2
BAB III IMPLEMENTASI PROGRAM.....3
3.1 Source Code Program.....3
3.2 Output Program.....6
3.3 Penjelasan Program.....7
BAB IV ANALISIS DAN PEMBAHASAN.....9
4.1 Analisis Hasil Metode Biseksi.....9
4.2 Analisis Hasil Metode Regula Falsi.....10
4.3 Perbandingan Kedua Metode.....11
BAB V PENUTUP.....13
5.1 Kesimpulan.....13
DAFTAR PUSTAKA.....14

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam dunia komputasi ilmiah, banyak permasalahan matematika yang tidak dapat diselesaikan secara analitik, khususnya persamaan non-linear. Oleh karena itu, diperlukan metode numerik untuk mencari solusi pendekatan terhadap akar suatu fungsi. Metode **Biseksi** dan **Regula Falsi** merupakan dua teknik dasar dalam menentukan akar persamaan yang memanfaatkan konsep iterasi dan interval, di mana fungsi mengalami perubahan tanda.

1.2 Tujuan

- Memahami prinsip dasar metode Biseksi dan Regula Falsi.
- Mengimplementasikan metode tersebut menggunakan bahasa Python.
- Membandingkan hasil dan tingkat konvergensi kedua metode.

1.3 Rumusan Masalah

- Bagaimana prinsip kerja metode Biseksi dan Regula Falsi?
- Bagaimana implementasi kedua metode dalam bahasa Python?
- Bagaimana perbandingan hasil dan tingkat konvergensi dari kedua metode tersebut?

BAB II
DASAR TEORI

2.1 Metode Biseksi

Metode Biseksi adalah salah satu teknik numerik untuk mencari akar persamaan f(x)=0 dengan cara membagi dua interval [a, b] secara berulang, Syarat utama metode ini adalah nilai fungsi di batas bawah dan atas memiliki tanda yang berbeda:

$f(a) \times f(b) < 0$

Rumus penentuan titik tengah:

$c = 2a + b$

Kemudian interval baru ditentukan berdasarkan tanda f(c). Proses diulang hingga galat (error) berada di bawah batas toleransi.

2.2 Metode Regula Falsi

Metode Regula Falsi atau *False Position Method* merupakan pengembangan dari metode Biseksi, Alih-alih mengambil titik tengah, metode ini menggunakan interpolasi linier antara dua titik batas:

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

Metode ini umumnya lebih cepat daripada Biseksi karena memperkirakan akar di posisi yang lebih akurat pada setiap iterasi, tetapi terkadang konvergensinya melambat jika salah satu batas tetap tidak berubah.

BAB III
IMPLEMENTASI PROGRAM

3.1 Source Code Program

a) Program metode biseksi

```
import pandas as pd


def metode_biseksi(fungsi, batas_bawah, batas_atas, toleransi=1e-6,
iterasi_maks=100):
      if fungsi(batas_bawah) * fungsi(batas_atas) > 0:
            print("Tanda fungsi pada interval [a, b] sama. Tidak dapat diterapkan
            metode Biseksi.")
      return None

      hasil_iterasi = []

      for langkah in range(1, iterasi_maks + 1):
            titik_tengah = (batas_bawah + batas_atas) / 2
            nilai_tengah = fungsi(titik_tengah)
            galat = abs(batas_atas - batas_bawah)

      hasil_iterasi.append([
            langkah,
            round(batas_bawah, 6),
            round(batas_atas, 6),
            round(titik_tengah, 8),
            round(nilai_tengah, 10),
            round(galat, 10)
      ])

      if abs(nilai_tengah) < toleransi or galat < toleransi:
            print(f"Akar ditemukan setelah {langkah} iterasi.")
            break

      if fungsi(batas_bawah) * nilai_tengah < 0:
            batas_atas = titik_tengah
            else:
            batas_bawah = titik_tengah
```

```
    tabel_hasil = pd.DataFrame(
    hasil_iterasi,
    columns=['Iterasi ke-', 'Batas Bawah (a)', 'Batas Atas (b)', 'Perkiraan Akar
    (c)', 'f(c)', 'Galat (|b - a|)']
    )

    print("\nTabel Proses Iterasi Metode Biseksi:\n")
    print(tabel_hasil.to_string(index=False))
    return titik_tengah


def fungsi(x):
    return 2*x**2 + 3*x - 4

batas_bawah = -4
batas_atas = 0

akar = metode_biseksi(fungsi, batas_bawah, batas_atas)

print("\nAkar dari f(x) = 0 adalah:", akar)
```

b) Program metode regula falsi

```
import pandas as pd

def regula_falsi(fungsi, batas_bawah, batas_atas, toleransi=1e-6,
iterasi_maks=100):
    if fungsi(batas_bawah) * fungsi(batas_atas) > 0:
        print("Tanda fungsi pada interval [a, b] sama. Tidak dapat diterapkan
        metode Regula Falsi.")
        return None

    hasil_iterasi = []

    c_sebelumnya = None
    for langkah in range(1, iterasi_maks + 1):
        titik_tengah = batas_atas - (fungsi(batas_atas) * (batas_atas -
        batas_bawah)) / (fungsi(batas_atas) - fungsi(batas_bawah))
        nilai_tengah = fungsi(titik_tengah)
            if c_sebelumnya is None:
                galat = None
            else:
```

```
                        galat = abs(titik_tengah - c_sebelumnya)
                        hasil_iterasi.append([
                              langkah,
                              round(batas_bawah, 6),
                              round(batas_atas, 6),
                              round(titik_tengah, 8),
                              round(nilai_tengah, 10),
                              None if galat is None else round(galat, 10)
                        ])

      if abs(nilai_tengah) < toleransi:
            print(f"Akar ditemukan setelah {langkah} iterasi.")
            break

      if fungsi(titik_tengah) * fungsi(batas_bawah) < 0:
            batas_atas = titik_tengah
      else:
            batas_bawah = titik_tengah
            c_sebelumnya = titik_tengah

      tabel_hasil = pd.DataFrame(
      hasil_iterasi,
      columns=['Iterasi ke-', 'Batas Bawah (a)', 'Batas Atas (b)', 'Perkiraan Akar
      (c)', 'f(c)', 'Galat (|cₙ - cₙ₋₁|)']
      )

      print("\nTabel Proses Iterasi Regula Falsi:\n")
      print(tabel_hasil.to_string(index=False))
      return titik_tengah

def fungsi(x):
      return 2*x**2 + 3*x - 4

batas_bawah = -4
batas_atas = 0

akar = regula_falsi(fungsi, batas_bawah, batas_atas)

print("\nAkar dari f(x) = 0 adalah:", akar)
```

3.2 Output Program

a) Metode Biseksi

```
● ➜ 0s ◎ python biseksi.py
Akar ditemukan setelah 23 iterasi.

Tabel Proses Iterasi Metode Biseksi:

 Iterasi ke-  Batas Bawah (a)  Batas Atas (b)  Perkiraan Akar (c)          f(c)  Galat (|b - a|)
           1       -4.000000        0.000000          -2.000000 -2.000000e+00     4.000000e+00
           2       -4.000000       -2.000000          -3.000000  5.000000e+00     2.000000e+00
           3       -3.000000       -2.000000          -2.500000  1.000000e+00     1.000000e+00
           4       -2.500000       -2.000000          -2.250000 -6.250000e-01     5.000000e-01
           5       -2.500000       -2.250000          -2.375000  1.562500e-01     2.500000e-01
           6       -2.375000       -2.250000          -2.312500 -2.421875e-01     1.250000e-01
           7       -2.375000       -2.312500          -2.343750 -4.492188e-02     6.250000e-02
           8       -2.375000       -2.343750          -2.359375  5.517578e-02     3.125000e-02
           9       -2.359375       -2.343750          -2.351562  5.004883e-03     1.562500e-02
          10       -2.351562       -2.343750          -2.347656 -1.998901e-02     7.812500e-03
          11       -2.351562       -2.347656          -2.349609 -7.499695e-03     3.906250e-03
          12       -2.351562       -2.349609          -2.350586 -1.249313e-03     1.953125e-03
          13       -2.351562       -2.350586          -2.351074  1.877308e-03     9.765625e-04
          14       -2.351074       -2.350586          -2.350830  3.138781e-04     4.882812e-04
          15       -2.350830       -2.350586          -2.350708 -4.677474e-04     2.441406e-04
          16       -2.350830       -2.350708          -2.350769 -7.694210e-05     1.220703e-04
          17       -2.350830       -2.350769          -2.350800  1.184661e-04     6.103520e-05
          18       -2.350800       -2.350769          -2.350784  2.076150e-05     3.051760e-05
          19       -2.350784       -2.350769          -2.350777 -2.809040e-05     1.525880e-05
          20       -2.350784       -2.350777          -2.350780 -3.664500e-06     7.629400e-06
          21       -2.350784       -2.350780          -2.350782  8.548500e-06     3.814700e-06
          22       -2.350782       -2.350780          -2.350781  2.442000e-06     1.907300e-06
          23       -2.350781       -2.350780          -2.350781 -6.112000e-07     9.537000e-07

Akar dari f(x) = 0 adalah: -2.350780963897705
```

b) Metode Regula Falsi

```
● ➜ 0s ◎ python regula-falsi.py
Akar ditemukan setelah 17 iterasi.

Tabel Proses Iterasi Regula Falsi:

 Iterasi ke-  Batas Bawah (a)  Batas Atas (b)  Perkiraan Akar (c)          f(c)  Galat (|cₙ - cₙ₋₁|)
           1              -4        0.000000          -0.800000 -5.120000e+00                  NaN
           2              -4       -0.800000          -1.575758 -3.761249e+00         7.757576e-01
           3              -4       -1.575758          -2.037175 -1.811362e+00         4.614171e-01
           4              -4       -2.037175          -2.236788 -7.039217e-01         1.996135e-01
           5              -4       -2.236788          -2.311092 -2.509842e-01         7.430369e-02
           6              -4       -2.311092          -2.337176 -8.674593e-02         2.608391e-02
           7              -4       -2.337176          -2.346142 -2.965892e-02         8.966588e-03
           8              -4       -2.346142          -2.349202 -1.010306e-02         3.060055e-03
           9              -4       -2.349202          -2.350244 -3.437180e-03         1.041724e-03
          10              -4       -2.350244          -2.350598 -1.168867e-03         3.543307e-04
          11              -4       -2.350599          -2.350719 -3.974335e-04         1.204869e-04
          12              -4       -2.350719          -2.350760 -1.351270e-04         4.096640e-05
          13              -4       -2.350760          -2.350774 -4.594230e-05         1.392840e-05
          14              -4       -2.350774          -2.350779 -1.562000e-05         4.735600e-06
          15              -4       -2.350779          -2.350780 -5.310600e-06         1.610000e-06
          16              -4       -2.350780          -2.350781 -1.805600e-06         5.474000e-07
          17              -4       -2.350781          -2.350781 -6.139000e-07         1.861000e-07

Akar dari f(x) = 0 adalah: -2.350780963486787
```

3.3 Penjelasan Program

Program ini dibuat menggunakan bahasa Python untuk mengimplementasikan dua metode pencarian akar fungsi non-linear, yaitu Metode Biseksi dan Metode Regula Falsi. Kedua metode ini digunakan untuk mencari akar dari fungsi:

$f(x)=2x2+3x−4$

Proses dilakukan secara iteratif hingga nilai yang diperoleh memenuhi batas toleransi galat (error) yang telah ditentukan.

a) Metode Biseksi

Program dimulai dengan mengimpor library pandas untuk menampilkan hasil iterasi dalam bentuk tabel.

Fungsi metode_biseksi() menerima parameter berupa fungsi, batas bawah, batas atas, toleransi, dan jumlah iterasi maksimum.

Langkah-langkah utamanya:

- Cek tanda fungsi memastikan $f(a)\times f(b)<0$ agar ada akar di dalam interval.
- Hitung titik tengah menggunakan rumus $c=(a+b)/2$.
- Hitung nilai fungsi dan galat pada setiap iterasi.
- Perbarui interval tergantung tanda fungsi di titik tengah.
- Simpan hasil iterasi ke dalam tabel DataFrame untuk ditampilkan.

Iterasi berhenti jika nilai fungsi sudah mendekati nol atau galat lebih kecil dari toleransi. Metode ini stabil dan selalu konvergen, tetapi relatif lebih lambat dibanding metode lain.

b) Metode Regula Falsi

Struktur dasarnya hampir sama dengan Biseksi, namun cara menghitung titik akar berbeda. Nilai perkiraan akar dihitung dengan interpolasi linier:

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

Metode ini sering kali lebih cepat karena memperkirakan akar lebih dekat ke posisi sebenarnya.

- Langkah utama:
- Tentukan titik akar baru menggunakan rumus di atas.
- Hitung nilai fungsi f(c) dan galat antar iterasi |cn−cn−1|.
- Perbarui batas interval sesuai tanda fungsi.
- Simpan hasil ke dalam tabel seperti pada Biseksi.

Metode Regula Falsi cenderung konvergen lebih cepat di awal, namun terkadang melambat jika salah satu batas tidak berubah.

BAB IV
ANALISIS DAN PEMBAHASAN

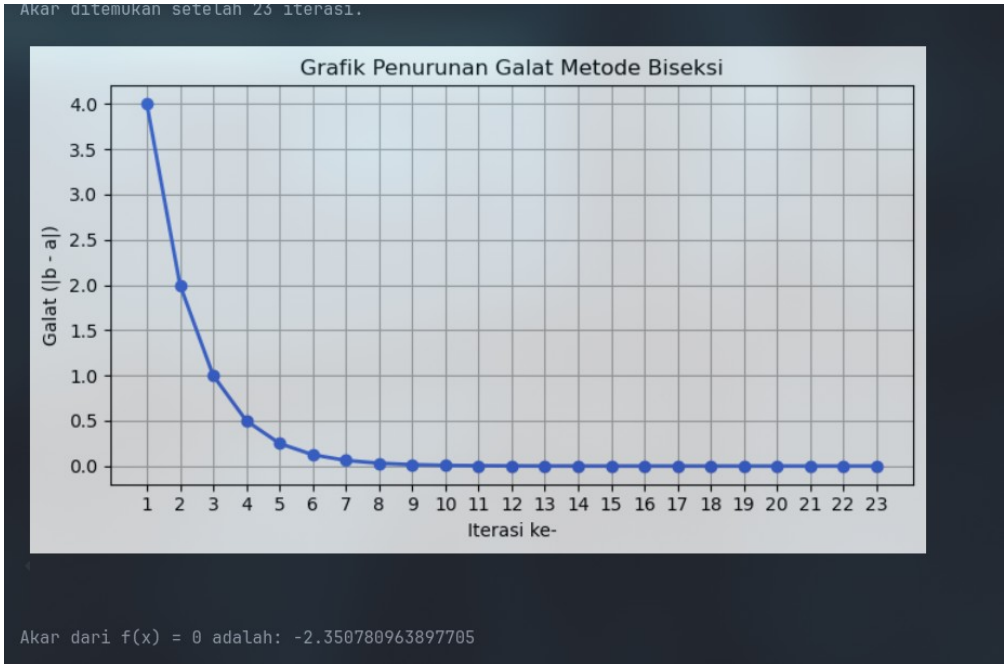
4.1 Analisis Hasil Metode Biseksi

Berdasarkan hasil eksekusi program Metode Biseksi, diperoleh bahwa nilai akar dari fungsi f(x)=2x2+3x−4 terletak di sekitar x = -2.35078, setelah melalui 23 iterasi.

Proses pencarian akar dilakukan dengan cara membagi dua interval [a, b] secara bertahap, hingga nilai galat |b−a| menjadi lebih kecil dari batas toleransi yang telah ditentukan (10−6).

Dari tabel hasil iterasi, terlihat bahwa setiap langkah menghasilkan interval yang semakin sempit, menandakan metode ini konvergen secara monoton menurun terhadap galatnya. Nilai f(c) juga semakin mendekati nol pada setiap iterasi, yang berarti bahwa hasil pendekatan semakin dekat dengan akar sebenarnya.

Metode Biseksi memiliki karakteristik konvergensi yang lambat namun sangat stabil. Hal ini disebabkan karena setiap iterasi hanya mengurangi setengah dari interval sebelumnya, sehingga pendekatannya pasti menuju ke akar tetapi memerlukan lebih banyak langkah dibanding metode lain.



Grafik penurunan galat pada menunjukkan bentuk kurva menurun secara eksponensial, di mana nilai galat |b−a| turun drastis pada beberapa iterasi pertama, lalu perlahan melandai mendekati nol.

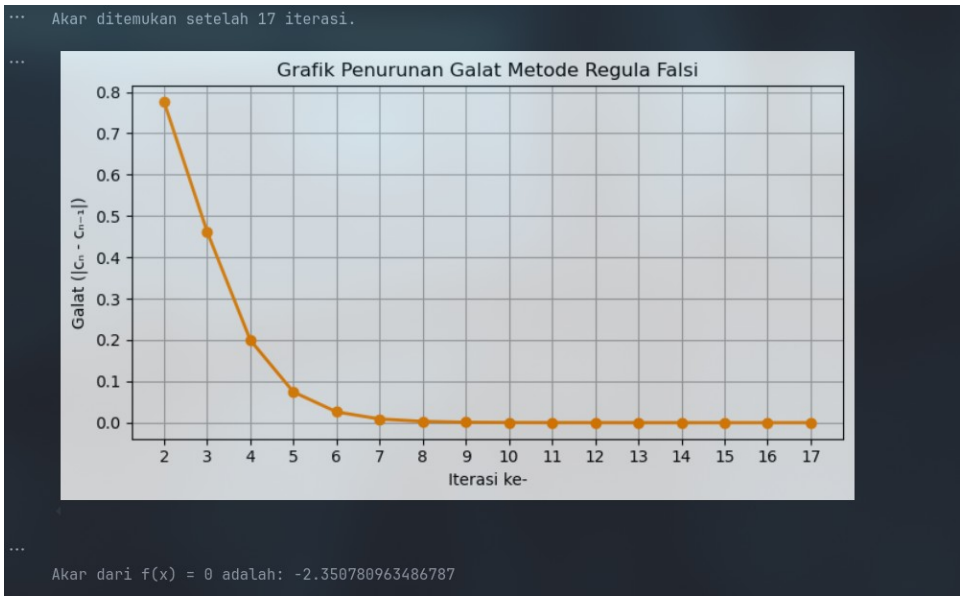
Hal ini menggambarkan bahwa metode Biseksi memiliki konvergensi linear, yaitu kecepatan konvergensi yang tetap dari iterasi ke iterasi.

Meskipun tidak secepat metode lain, kestabilan hasilnya menjadikan Biseksi sebagai metode dasar yang pasti konvergen asalkan fungsi memenuhi syarat perubahan tanda dalam interval awal.

4.2 Analisis Hasil Metode Regula Falsi

Pada metode Regula Falsi, akar persamaan yang sama juga ditemukan di sekitar x = -2.35078, namun hanya membutuhkan 17 iterasi untuk mencapai toleransi error yang sama. Perbedaan jumlah iterasi ini terjadi karena metode Regula Falsi menggunakan interpolasi linier, bukan pembagian dua seperti Biseksi.

Secara prinsip, metode ini mencari titik potong garis lurus antara dua titik (a,f(a)) dan (b,f(b)), sehingga perkiraan akar c lebih dekat ke akar sebenarnya pada iterasi awal. Hal ini menyebabkan metode Regula Falsi memiliki konvergensi lebih cepat di tahap awal.



Dari grafik penurunan galat, terlihat bahwa nilai galat |cn−cn−1| turun tajam pada awal iterasi, lalu cenderung melandai menjelang akhir proses.

Pola ini menunjukkan bahwa Regula Falsi memiliki konvergensi super-linear di awal, namun dapat stagnan apabila salah satu batas interval (a atau b) tidak berubah secara signifikan. Stagnasi ini biasanya terjadi ketika fungsi pada salah satu batas sudah sangat kecil, sehingga garis interpolasi hampir sejajar dengan sumbu X.

Dengan demikian, walaupun Regula Falsi lebih cepat mencapai akar dibanding Biseksi, konvergensinya tidak selalu konsisten di seluruh proses iterasi.

Namun secara umum, hasil akhirnya tetap memiliki akurasi yang sangat baik dan mendekati nilai akar sebenarnya.

4.3 Perbandingan Kedua Metode

Aspek	Metode Biseksi	Metode Regula Falsi
Prinsip Dasar	Membagi dua interval secara iteratif	Interpolasi linier antara dua titik
Kecepatan Konvergensi	Lebih lambat (konvergensi linear)	Lebih cepat di awal (super-linear)
Stabilitas Hasil	Sangat stabil, selalu konvergen	Bisa stagnan di akhir iterasi
Kompleksitas	Lebih sederhana dan mudah diterapkan	Lebih kompleks karena perhitungan interpolasi
Ketergantungan pada Fungsi	Tidak tergantung bentuk fungsi	Efektif untuk fungsi hampir linear
Jumlah Iterasi	Lebih banyak (23 iterasi)	Lebih sedikit (17 iterasi)
Akurasi Akhir	Tinggi dan konsisten	Sama tinggi, tapi tergantung kondisi batas

Berdasarkan hasil perbandingan di atas, kedua metode menghasilkan akar yang identik dengan perbedaan sangat kecil pada digit desimal terakhir, Namun dari segi efisiensi, metode Regula Falsi lebih unggul karena membutuhkan iterasi lebih sedikit untuk mencapai galat yang sama,

Sementara itu, metode Biseksi lebih unggul dalam hal kestabilan karena tidak bergantung pada bentuk fungsi dan selalu konvergen selama syarat perubahan tanda terpenuhi. Pada grafik biseksi dan regula falsi perbedaan karakter konvergensi terlihat jelas:

- Kurva Biseksi menurun halus dan teratur (stabil, lambat).
- Kurva Regula Falsi menurun tajam di awal, lalu melambat di akhir (cepat tapi tidak selalu stabil).

BAB V
PENUTUP

5.1 Kesimpulan

Metode Biseksi dan Metode Regula Falsi merupakan dua pendekatan numerik yang digunakan untuk mencari akar persamaan non-linear secara iteratif. Berdasarkan hasil implementasi menggunakan bahasa Python, keduanya terbukti mampu menghasilkan nilai akar yang sama, yaitu sekitar $x=-2.35078$ untuk fungsi $f(x)=2x2+3x-4$, dengan tingkat akurasi yang tinggi. Metode Biseksi memiliki karakteristik konvergensi yang lambat namun stabil karena interval pencarian selalu dibagi dua pada setiap iterasi, sehingga galat menurun secara teratur hingga mencapai batas toleransi.

Sementara itu, Metode Regula Falsi menggunakan pendekatan interpolasi linier sehingga mampu memperkirakan akar lebih cepat di awal iterasi, meskipun terkadang mengalami perlambatan konvergensi di tahap akhir ketika salah satu batas interval tidak berubah. Berdasarkan hasil analisis, dapat disimpulkan bahwa Metode Biseksi lebih unggul dalam hal kestabilan dan kepastian konvergensi, sedangkan Metode Regula Falsi lebih efisien dari segi kecepatan iterasi. Oleh karena itu, pemilihan metode sebaiknya disesuaikan dengan kebutuhan: Metode Biseksi digunakan ketika kestabilan hasil menjadi prioritas, sedangkan Regula Falsi dipilih ketika efisiensi waktu perhitungan lebih diutamakan. Secara keseluruhan, kedua metode ini menunjukkan bahwa pendekatan numerik dapat menjadi solusi efektif untuk menentukan akar fungsi yang tidak dapat diselesaikan secara analitik, serta memberikan dasar penting dalam penerapan komputasi ilmiah di bidang teknik dan sains.

DAFTAR PUSTAKA

Munir, R. (2019). *Metode Numerik*. Bandung: Informatika Bandung.

Kadir, A. (2016). *Dasar Pemrograman Python untuk Ilmu Komputer dan Rekayasa*. Yogyakarta: ANDI.

Suhardi, D. (2020). *Analisis Metode Numerik dan Implementasinya dengan Python*. Yogyakarta: Deepublish.