

# LAPORAN PRAKTIKUM 3

## METODE TITIK TETAP & NEWTON RAPHSON



Disusun oleh:

Nama : Muhammad Fathurrahman  
NIM : 2024573010004  
Kelas/ Semester : TI 2C/ 3  
Mata Kuliah : Praktikum Metode Numerik  
No. Prak : 04/PMN/TI/2025  
Program Studi : Teknik Informatika

**TEKNOLOGI INFORMASI DAN KOMPUTER**  
**POLITEKNIK NEGERI LHOKSEUMAWE**  
**2025**

## **LEMBAR PENGESAHAN**

Nomor Praktikum : 03/PNM/TI/2025  
Nama Praktikum : Laporan Praktikum 3 Titik Tetap & Newton  
Nama Praktika : Raphson  
NIM : 2024573010004  
Kelas : TI 2C  
Jurusan : Teknologi Informasi dan Komputer  
Prodi : Teknik Informatika  
Tanggal Praktikum : 12 November 2025  
Tanggal Penyerahan : 17 November 2025  
Nilai :  
Keterangan :

Buket Rata, 17 November 2025

Dosen Pembimbing,

**Radhiyatammardhiyyah, SST., M.Sc.**

**NIP. 199208262022032011**

## KATA PENGANTAR

Assalamualaikum warahmatullahi wabarakatuh

Puji syukur ke hadirat Tuhan Yang Maha Esa. Atas rahmat dan hidayah NYA, penulis dapat menyelesaikan Tugas Praktikum tentang “Metode Titik Tetap & Newton Raphson” dengan tepat waktu.

Makalah disusun untuk memenuhi tugas Mata Kuliah Praktikum Metode Numerik. Selain itu, makalah ini bertujuan menambah wawasan tentang Metode Titik Tetap & Newton Raphson jika diterapkan pada Bahasa pemrograman Python

Penulis mengucapkan terima kasih kepada Ibu Radhiyatammardhiyyah selaku guru Mata Pelajaran Sejarah. Ucapan terima kasih juga disampaikan kepada semua pihak yang telah membantu diselesaiannya makalah ini.

Penulis menyadari makalah ini masih jauh dari sempurna. Oleh sebab itu, saran dan kritik yang membangun diharapkan demi kesempurnaan makalah ini.

Wassalamualaikum warahmatullahi wabarakatuh

Lhokseumawe, 12 November 2025

Muhammad Fathurrahman

## DAFTAR ISI

<b>LEMBAR PENGESAHAN .....</b>	<b>i</b>
<b>KATA PENGANTAR.....</b>	<b>ii</b>
<b>DAFTAR ISI.....</b>	<b>iii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>4</b>
1.1    Dasar Teori.....	4
<b>BAB 2 PRAKTIKUM .....</b>	<b>5</b>
2.1    Metode Titik Tetap.....	5
2.2    Metode Newton Raphson.....	9
<b>BAB 3 KESIMPULAN .....</b>	<b>11</b>
<b>DAFTAR PUSTAKA.....</b>	<b>12</b>

# BAB 1

## PENDAHULUAN

### 1.1 Dasar Teori

Metode Titik Tetap adalah salah satu metode numerik untuk mencari solusi persamaan non-linear dengan cara mengubah persamaan  $f(x)=0$  menjadi bentuk  $x=g(x)$ . Ide utama metode ini adalah memilih suatu fungsi  $g(x)$  yang memenuhi syarat konvergensi, kemudian melakukan iterasi  $x_{r+1} = g(x_r)$  hingga nilai yang dihasilkan mendekati solusi. Konvergensi metode titik tetap sangat dipengaruhi oleh bentuk fungsi  $g(x)$  serta nilai awal  $x_0$ . Jika turunan  $|g'(x)| < 1$  pada sekitar titik akar, maka iterasi akan cenderung konvergen. Metode ini sederhana dan mudah diterapkan, namun kadang lambat dan sangat bergantung pada pemilihan bentuk  $g(x)$  yang tepat.

Metode Newton-Raphson adalah metode numerik yang lebih cepat untuk mencari akar persamaan non-linear dengan memanfaatkan turunan pertama dari fungsi. Metode ini didasarkan pada pendekatan garis singgung terhadap kurva fungsi di titik tertentu. Rumus iterasinya adalah  $x_{r+1} = x_r - \frac{f(x)}{f'(x)}$ . Dengan menggunakan informasi kemiringan dari turunan, metode ini biasanya memiliki tingkat konvergensi kuadratik, sehingga lebih cepat mencapai akar dibandingkan metode titik tetap. Namun, metode Newton-Raphson membutuhkan perhitungan turunan dan dapat gagal jika nilai awal terlalu jauh dari akar atau jika turunan mendekati nol pada titik iterasi.

## BAB 2

# PRAKTIKUM

### 2.1 Metode Titik Tetap

Mencari nilai akar dari persamaan  $x^2 - 2x - 3 = 0$  dengan nilai error = 0.000001.

**Penyelesaian:**

$$a) \quad x^2 - 2x - 3 = 0$$

$$x^2 = 2x + 3$$

$$x = \sqrt{2x + 3}$$

Tebakan awal  $x_0 = 4$

Code:

```
import math

def xr(x):
    return math.sqrt(2*x + 3)

def titik_tetap(x0, toleransi, max_iter=100):
    print("r\t x_r\t |xr+1-xr|\t galat")
    for i in range(max_iter):
        x1 = xr(x0)
        error = abs(x1 - x0)
        print(f"{i}\t{x0:.6f}\t{x1:.6f}\t{error:.6f}\t")
        if error < toleransi:
            print("\nHasil Titik Tetap = ", x1, "\n")
            return x1
        x0 = x1
    print("===== TITIK TETAP =====")
    titik_tetap(4, 1e-6)
```

Output:

===== TITIK TETAP =====			
r	x_r	xr+1-xr	galat
0	4.000000	3.316625	0.683375
1	3.316625	3.103748	0.212877
2	3.103748	3.034385	0.069362
3	3.034385	3.011440	0.022945
4	3.011440	3.003811	0.007629
5	3.003811	3.001270	0.002541
6	3.001270	3.000423	0.000847
7	3.000423	3.000141	0.000282
8	3.000141	3.000047	0.000094
9	3.000047	3.000016	0.000031
10	3.000016	3.000005	0.000010
11	3.000005	3.000002	0.000003
12	3.000002	3.000001	0.000001
13	3.000001	3.000000	0.000000

Analisis:

$$b) \quad x^2 - 2x - 3 = 0$$

$$x(x - 2) = 3$$

$$x = \frac{3}{x - 2}$$

Tebakan awal  $x_0 = 4$

Code:

```
def xr(x):
    return (3/(x-2))

def titik_tetap(x0, toleransi, max_iter=100):
    print("r\t x_r\t|xr+1-xr|\tgat")
    for i in range(max_iter):
        x1 = xr(x0)
        galat = abs(x1 - x0)
        print(f"{i}\t{x0:.6f}\t{x1:.6f}\t{galat:.6f}")
        if galat < toleransi:
            print("\nHasil Titik Tetap = {x1:.3f} \n")
            return x1
        x0 = x1

print("===== TITIK TETAP =====")
titik_tetap(4, 1e-6)
```

Output:

===== TITIK TETAP =====			
r	x_r	xr+1-xr	galat
0	4.000000	1.500000	2.500000
1	1.500000	-6.000000	7.500000
2	-6.000000	-0.375000	5.625000
3	-0.375000	-1.263158	0.888158
4	-1.263158	-0.919355	0.343803
5	-0.919355	-1.263158	0.888158
6	-1.263158	-0.919355	0.343803
7	-0.919355	-1.027624	0.108269
8	-1.027624	-0.990876	0.036748
9	-0.990876	-1.003051	0.012175
10	-1.003051	-0.998984	0.004066
11	-0.998984	-1.000339	0.001355
12	-1.000339	-0.999887	0.000452
13	-0.999887	-1.000038	0.000151
14	-1.000038	-0.999987	0.000050
15	-0.999987	-1.000004	0.000017
16	-1.000004	-0.999999	0.000006
	-0.999999	-1.000000	0.000002
	-1.000000	-1.000000	0.000001

Analisis:

$$c) \quad x^2 - 2x - 3 = 0$$

$$x = \frac{x^2 - 3}{2}$$

Tebakan awal  $x_0 = 4$

Code:

```
def xr(x):
    return ((x**2-3)/2)

def titik_tetap(x0, toleransi, max_iter=5):
    print("Iter\t x_r\t |xr+1-xr|\tgalat")
    for i in range(1, max_iter + 1):
        x1 = xr(x0)
        error = abs(x1 - x0)
        print(f"{i}\t{x0:.6f}\t{x1:.6f}\t{error:.6f}")
        if error < toleransi:
            print("\nHasil Titik Tetap =", x1, "\n")
            return x1
        x0 = x1
    print("===== TITIK TETAP =====")
titik_tetap(4, 1e-6)
```

Output:

```
===== TITIK TETAP =====
Iter      x_r          |xr+1-xr|      galat
1        4.000000      6.500000      2.500000
2        6.500000      19.625000     13.125000
3        19.625000     191.070312    171.445312
4        191.070312    18252.432159   18061.361847
5        18252.432159  166575638.367185  166557385.935025
```

Analisis:

$$d) \quad x^3 + 6x - 3 = 0$$

$$x = \frac{-x^3 + 3}{6}$$

Tebakan awal  $x_0 = 4$

Code:

```
def xr(x):
    return ((-x**3+3)/6)

def titik_tetap(x0, tolerasi=1e-5, max_iter=100):
    print("Iter\t x_r\t|xr+1-xr|\t galat\t")
    for i in range(1, max_iter + 1):
        x1 = xr(x0)
        galat = abs(x1 - x0)
        print(f"\t{i}\t{x0:.6f}\t{x1:.6f}\t{galat:.6f}\t")
        if galat < tolerasi:
            print("\nHasil Titik Tetap =", x1, "\n")
            return x1
        x0 = x1

print("===== TITIK TETAP =====")
titik_tetap(0.5)
```

Output:

===== TITIK TETAP =====			
Iter	x_r	xr+1-xr	galat
1	0.500000	0.479167	0.020833
2	0.479167	0.481664	0.002497
3	0.481664	0.481376	0.000288
4	0.481376	0.481409	0.000033
5	0.481409	0.481405	0.000004

Analisis:

## 2.2 Metode Newton Raphson

- a) Mencari nilai akar  $f(x) = e^x - 5x$  dengan metode Newton Raphson.

Menggunakan nilai error = 0.00001. Tebakan awal akar  $x_0 = 1$ .

**Penyelesaian:**

$$f(x) = e^x - 5x^2$$
$$f'(x) = e^x - 10x$$

Prosedur pelajaran Newton-Rapshon:

$$x_{r+1} = x_r - \frac{e^x - 5x^2}{e^x - 10x}$$

Tebakan awal  $x_0 = 0.5$

Code:

```
import math

def f(x):
    return math.exp(x) - (5*x**2)

def df(x):
    return math.exp(x) - 10*x

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:.6f} | {abs(x1-x0):.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(0.5, 1e-5, 100)
print("Akar hampiran =", akar)
```

Output:

r	x_n	x1-x0
1	0.618976	0.118976
2	0.605444	0.013532
3	0.605267	0.000177
4	0.605267	0.000000

Akar hampiran = 0.6052671213146193

Analisis:

- b) Mencari nilai akar dari  $f(x) = 2x^2 + 3x - 4$ .  $e = 0.001$  dengan  $x_0 = 1$ .

Code:

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

def df(x):
    return 4*x + 3

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:.6f} | {abs(x1-x0):.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(1, 1e-3, 100)
print("Akar hampiran =", akar)
```

Output:

===== NEWTON RAPHSON =====		
r	x_n	x1-x0
1	0.857143	0.142857
2	0.850794	0.006349
3	0.850781	0.000013

Akar hampiran = 0.8507810594077327

Analisis:

c) Mencari nilai akar dari  $f(x) = x^3 - 2x - 5$ .  $\epsilon = 0.001$  dengan  $x_0 = 2$ .

Code:

```
import math

def f(x):
    return math.cos(x) - x

def df(x):
    return -math.sin(x) - 1

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:12.6f} | {abs(x1-x0):12.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(1, 1e-4, 100)
print("Akar hampiran =", akar)
```

Output:

```
===== NEWTON RAPHSON =====
r | x_n | x1-x0
1 | 2.100000 | 0.100000
2 | 2.094568 | 0.005432
3 | 2.094551 | 0.000017
Akar hampiran = 2.094551481698199
```

Analisis:

- d) Mencari nilai akar dari  $f(x) = \cos(x) - x$ .  $\epsilon = 0.0001$  dengan  $x_0 = 1$ .

Code:

```
def f(x):
    return x**3 - 2*x - 5

def df(x):
    return 3*x**2 - 2

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:12.6f} | {abs(x1-x0):12.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(2, 1e-3, 100)
print("Akar hampiran =", akar)
```

Output:

```
===== NEWTON RAPHSON =====
r | x_n | x1-x0
1 | 0.750364 | 0.249636
2 | 0.739113 | 0.011251
3 | 0.739085 | 0.000028
Akar hampiran = 0.739085133385284
```

Analisis:

- e) Mencari nilai akar dari  $f(x) = e^x + x^2 - 3$ .  $\epsilon = 0.0001$  dengan  $x_0 = 0$ .

Code:

```
import math

def f(x):
    return math.exp(x) + x**2 - 3

def df(x):
    return math.exp(x) + 2*x

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:12.6f} | {abs(x1-x0):12.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(0, 1e-4, 100)
print("Akar hampiran =", akar)
```

Output:

```
===== NEWTON RAPHSON =====
r | x_n | x1-x0
1 | 2.000000 | 2.000000
2 | 1.263411 | 0.736589
3 | 0.911568 | 0.351842
4 | 0.837536 | 0.074032
5 | 0.834492 | 0.003044
6 | 0.834487 | 0.000005
Akar hampiran = 0.8344868653224407
```

Analisis:

f) Mencari nilai akar dari  $f(x) = x^3 - 9x + 3$ .  $e = 0.001$  dengan  $x_0 = 0.5$ .

Code:

```
import math

def f(x):
    return x**3 - 9*x + 3

def df(x):
    return 3*x**2 - 9

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:.6f} | {abs(x1-x0):.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(0.5, 1e-3, 100)
print("Akar hampiran =", akar)
```

Output:

```
===== NEWTON RAPHSON =====
r | x_n | x1-x0
1 | 0.333333 | 0.166667
2 | 0.337607 | 0.004274
3 | 0.337609 | 0.000002
Akar hampiran = 0.3376089559653128
```

Analisis:

g) Mencari nilai akar dari  $f(x) = \sin(x) - 0.5$ .  $e = 0.001$  dengan  $x_0 = 1$ .

Code:

```
import math

def f(x):
    return math.sin(x) - 0.5

def df(x):
    return math.cos(x)

def newton_raphson(x0, tol, max_iter):
    print("r | x_n | x1-x0 ")
    for i in range(max_iter):
        fx = f(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Turunan nol. Tidak dapat melanjutkan iterasi.")
            return None
        x1 = x0 - fx / dfx
        print(f"{i+1} | {x1:.6f} | {abs(x1-x0):.6f}")
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    return None

print("===== NEWTON RAPHSON =====")
akar = newton_raphson(1, 1e-4, 100)
print("Akar hampiran =", akar)
```

Output:

```
===== NEWTON RAPHSON =====
r | x_n | error
1 | 0.368000 | 0.632000
2 | 0.518314 | 0.150314
3 | 0.523591 | 0.005277
4 | 0.523599 | 0.000008
Akar hampiran = 0.5235987755798704
```

Analisis:

## **BAB 3**

### **KESIMPULAN**

Secara keseluruhan, metode Titik Tetap dan Newton-Raphson merupakan dua teknik penting dalam penyelesaian persamaan non-linear, masing-masing dengan kelebihan dan kekurangannya. Metode Titik Tetap lebih sederhana dan tidak memerlukan turunan, namun tingkat konvergensi lambat dan sangat bergantung pada pemilihan fungsi iterasi yang sesuai. Sebaliknya, metode Newton-Raphson menawarkan konvergensi yang jauh lebih cepat melalui penggunaan turunan pertama, tetapi membutuhkan nilai awal yang cukup dekat dengan akar dan perhitungan turunan yang akurat. Dengan memahami karakteristik masing-masing metode, pengguna dapat memilih teknik yang paling efektif sesuai dengan tipe permasalahan yang dihadapi.

## **DAFTAR PUSTAKA**

Ma'arif, A. 2020. Buku Ajar Pemrograman Lanjut Bahasa Pemrograman Python. Yogyakarta: Universitas Ahmad Dahlan.