International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Comparative Study of Numerical Methods for Estimating Effective Interest Rate in Household Instalment Plans

Kejia[1], Dr. Vinod Kumar Sharma[2]

[1]Research Scholar, Department of Mathematics, Tantia University, Sri Ganganagar
[2]Professor, Department of Mathematics, Tantia University, Sri Ganganagar

Abstract
Numerical methods are useful for solving equations that arise from real-life problems when exact algebraic solutions are difficult. One such real-life problem is the estimation of the effective interest rate in household instalment plans. In many household purchases, the consumer knows the cash price, monthly instalment, and repayment period, but the actual monthly interest rate is not directly visible. When the interest rate is treated as the unknown variable, the equated monthly instalment equation becomes nonlinear. This paper compares four numerical methods, namely the bisection method, regula falsi method, secant method, and Newton-Raphson method, for estimating the monthly interest rate in household instalment plans. The dataset of twelve realistic household instalment cases are used. All four methods are solved using Python programming. The methods are compared on the basis of speed and accuracy using number of iterations, function evaluations, residual error, root error, and average execution time. The results show that Newton-Raphson required the fewest iterations, while the secant method gave very high accuracy without requiring derivative calculation. The bisection method was slower but reliable. The study shows that numerical methods can help consumers estimate hidden borrowing costs in instalment-based financial decisions.

Keywords: Numerical Methods, Nonlinear Equation, Bisection Method, Regula Falsi Method, Secant Method, Newton-Raphson Method, EMI, Effective Interest Rate, Household Instalment Plans

1. Introduction

Mathematics is not only a theoretical subject but also a practical tool for solving real-life problems. Many situations in finance, science, engineering, and commerce produce equations in which the required value cannot be found easily by direct algebraic methods. In such cases, numerical methods are used to obtain approximate solutions with a desired level of accuracy.

A common real-life financial problem is the estimation of the actual interest rate in household instalment plans. People often purchase items such as smartphones, refrigerators, laptops, washing machines, home inverters, medical services, or home repair services through fixed monthly instalments. In many such cases, the principal amount, monthly instalment, and repayment period are known, but the actual effective interest rate is not clearly shown to the consumer.

The equated monthly instalment equation contains the unknown interest rate both as a direct multiplier and inside a power term. Therefore, when the interest rate is unknown, the equation becomes nonlinear.

This makes the problem suitable for numerical root-finding methods.

The present study compares four numerical methods for solving this real-life nonlinear equation. The methods used are the bisection method, regula falsi method, secant method, and Newton-Raphson method. Python programming is used to solve the equation and compare the speed and accuracy of the methods.

2. Review of Literature

Abdulaziz G. Ahmad (2015) compared the bisection method and Newton-Raphson method for solving nonlinear root-finding problems. The study used MATLAB 7.6 and compared the convergence of both methods for a nonlinear equation. Ahmad reported that the Newton-Raphson method converged faster than the bisection method, which supports the need to compare numerical methods using iteration count and convergence speed.

Isaac Azure, Golbert Aloliga, and Louis Doabil (2020) compared five numerical methods: bisection, Newton-Raphson, regula falsi, secant, and fixed-point iteration. Their study focused on the rate of convergence of different numerical methods for solving nonlinear equations. The paper reported that all methods converged, but the number of iterations differed from method to method. This supports the present study's comparison of numerical methods using speed and accuracy measures.

Elsayed Badr, Sultan Almotairi, and Abdallah El Ghamry (2021) studied traditional and hybrid root-finding algorithms. Their paper emphasized that both the number of iterations and average running time are important criteria for evaluating numerical algorithms. This is directly related to the present study because the current research compares bisection, regula falsi, secant, and Newton-Raphson methods using both iteration count and programming speed.

Marcela Flórez, Miguel Vera, Juan Salazar-Torres, Yolanda Huérfano, Elkin Gelvez-Almeida, Oscar Valbuena, M. I. Vera, and M. Aranguen (2019) studied interest-rate calculation in certain ordinary annuities. Their work explained that calculating the interest rate in an annuity involves a non-analytical equation that requires numerical techniques. This study is important for the present research because household instalment plans also follow an annuity-type repayment structure where the interest rate must be estimated numerically.

Monica Noviyanti Tampubolon, Candra Ditasona, Jeny Siregar, and Rido Simbolon (2026) applied the Newton-Raphson method to compare effective interest rates in motorcycle instalment plans. They noted that consumers often receive information such as down payment, monthly instalment, and loan period without clear information about the effective interest rate. Their study supports the present research problem because the current paper also estimates hidden effective interest rates from instalment information.

From the above studies, it is clear that numerical methods are widely used for solving nonlinear equations and that interest-rate calculation in instalment or annuity problems often requires numerical techniques. However, many studies focus either on general nonlinear equations or on one financial method. The present study fills this gap by comparing four numerical methods through Python programming for household instalment plans.

3. Objectives of the Study

The main objectives of this study are:

1. To form a nonlinear equation from a real-life household instalment problem.
2. To estimate the unknown monthly interest rate using numerical methods.

3. To solve the equation using bisection, regula falsi, secant, and Newton-Raphson methods through Python programming.
4. To compare the speed of the methods using iteration count, function evaluations, and average execution time.
5. To compare the accuracy of the methods using residual error and root error.
6. To show the practical use of numerical methods in household financial decision-making.

4. Mathematical Formulation of the Real-Life Problem

Let:

P = principal amount or cash price
M = monthly instalment
n = number of monthly instalments
r = monthly interest rate

The standard EMI formula is:

$$M = \frac{Pr(1+r)^n}{(1+r)^n-1} \quad (1)$$

In this study, P, M, and n are known. The unknown variable is r. Therefore, the equation is rearranged as:

$$f(r) = \frac{Pr(1+r)^n}{(1+r)^n-1} - M = 0 \quad (2)$$

The root of this equation gives the monthly interest rate.
After estimating r, the effective annual rate is calculated as:

$$EAR = (1+r)^{12} - 1 \quad (3)$$

where EAR is the effective annual rate.
At $r = 0$, the EMI value is treated using the limiting value:

$$M = \frac{P}{n} \quad (4)$$

5. The Dataset

The following dataset was used for demonstration.

Table 1: Household instalment dataset

Case	Real-Life Use	Principal P in ₹	Tenure n months	Monthly EMI M in ₹	Total Paid in ₹	Extra Paid in ₹
C1	Student smartphone purchase	22,000	10	2,430	24,300	2,300

C2	Refrigerator purchase	36,000	12	3,338	40,056	4,056
C3	Student laptop purchase	62,000	12	5,935	71,220	9,220
C4	Washing machine purchase	48,000	18	3,132	56,376	8,376
C5	Sewing machine for home tailoring	75,000	18	4,959	89,262	14,262
C6	Dental treatment instalment	82,000	24	4,070	97,680	15,680
C7	Home inverter and battery	56,000	24	2,861	68,664	12,664
C8	Rooftop plumbing repair	98,000	30	4,110	123,300	25,300
C9	Home renovation materials	155,000	36	5,907	212,652	57,652
C10	Desktop computer for tutoring	90,000	24	4,731	113,544	23,544
C11	Kitchen appliance combo	68,000	15	5,312	79,680	11,680
C12	Medical treatment instalment	120,000	24	5,853	140,472	20,472

6. Numerical Methods Used

6.1 Bisection Method

The bisection method is a bracketing method. It starts with two values a and b, where:

$$f(a)f(b) < 0 \tag{5}$$

The midpoint is calculated as:

$$c = \frac{a+b}{2} \tag{6}$$

If f(c) is close to zero, c is accepted as the root. Otherwise, the interval is reduced by selecting the half interval where the sign change occurs.

6.2 Regula Falsi Method

The regula falsi method, also called the false position method, is another bracketing method. It estimates the root using a straight line between $(a, f(a))$ and $(b, f(b))$. The next approximation is:

$$c = \frac{af(b)-bf(a)}{f(b)-f(a)} \tag{7}$$

This method often converges faster than the bisection method because it uses function values to estimate the root position.

6.3 Secant Method

The secant method uses two initial approximations x_0 and x_1. The next approximation is calculated as:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} \quad (8)$$

The secant method does not require a derivative, which makes it useful when the derivative is difficult to calculate.

6.4 Newton-Raphson Method

The Newton-Raphson method uses the formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (9)$$

For the EMI function,

$$f(r) = \frac{Pr(1+r)^n}{(1+r)^n - 1} - M \quad (10)$$

the derivative used in the program is:

$$f'(r) = P\frac{(1+r)^n((1+r)^n - 1) - rn(1+r)^{n-1}}{((1+r)^n - 1)^2} \quad (11)$$

Newton-Raphson usually converges very fast when the initial guess is suitable.

7. Research Methodology

The research methodology followed these steps:

1. A real-life household instalment problem was selected.
2. The EMI equation was converted into a nonlinear equation.
3. Twelve unpublished author-generated household instalment cases were prepared.
4. The monthly interest rate was searched in the interval:

$$0 \le r \le 0.05 \quad (12)$$

This means the monthly interest rate was searched between 0% and 5%.

5. Four numerical methods were programmed in Python.
6. The stopping tolerance was fixed as 10^{-10}.
7. A high-precision bisection result with tolerance 10^{-15} was used as the reference root.
8. Speed was measured using iteration count, function evaluations, and average Python execution time.
9. Accuracy was measured using residual error and root error.

The residual error is: $|f(r)|$ (13)

The root error is: $|r_{method} - r_{reference}|$ (14)

8. Python Programming

```
from math import fabs
from time import perf_counter
from statistics import mean
```

```
TOL = 1e-10
A = 0.0
B = 0.05
REPEATS = 3000
data = [
("C1",  "Student smartphone purchase",       22000,  10, 2430),
("C2",  "Refrigerator purchase",             36000,  12, 3338),
("C3",  "Student laptop purchase",           62000,  12, 5935),
("C4",  "Washing machine purchase",          48000,  18, 3132),
("C5",  "Sewing machine for home tailoring", 75000,  18, 4959),
("C6",  "Dental treatment instalment",       82000,  24, 4070),
("C7",  "Home inverter and battery",         56000,  24, 2861),
("C8",  "Rooftop plumbing repair",           98000,  30, 4110),
("C9",  "Home renovation materials",         155000,  36, 5907),
("C10", "Desktop computer for tutoring",     90000,  24, 4731),
("C11", "Kitchen appliance combo",           68000,  15, 5312),
("C12", "Medical treatment instalment",      120000,  24, 5853),
]
def emi(P, r, n):
if fabs(r) < 1e-14:
return P / n
u = (1 + r) ** n
return P * r * u / (u - 1)

def f(P, M, n, r):
return emi(P, r, n) - M

def derivative(P, M, n, r):
if fabs(r) < 1e-8:
h = 1e-6
return (f(P, M, n, r + h) - f(P, M, n, r - h)) / (2 * h)

u = (1 + r) ** n
up = n * (1 + r) ** (n - 1)
return P * (u * (u - 1) - r * up) / ((u - 1) ** 2)

def result(method, root, iterations, evaluations, residual, history, converged):
return {
"method": method,
"root": root,
"iterations": iterations,
"evaluations": evaluations,
"residual": abs(residual),
```

```
"history": history,
"converged": converged
}

def bisection(P, M, n, a=A, b=B, tol=TOL, max_iter=300):
fa = f(P, M, n, a)
fb = f(P, M, n, b)
evaluations = 2
history = []

if fa * fb > 0:
return result("Bisection", None, 0, evaluations, float("nan"), history, False)

for i in range(1, max_iter + 1):
c = (a + b) / 2
fc = f(P, M, n, c)
evaluations += 1
history.append((i, c, fc))

if abs(fc) < tol or abs(b - a) / 2 < tol:
return result("Bisection", c, i, evaluations, fc, history, True)

if fa * fc < 0:
b = c
fb = fc
else:
a = c
fa = fc

return result("Bisection", c, max_iter, evaluations, fc, history, False)

def regula_falsi(P, M, n, a=A, b=B, tol=TOL, max_iter=300):
fa = f(P, M, n, a)
fb = f(P, M, n, b)
evaluations = 2
history = []
c_old = None

if fa * fb > 0:
return result("Regula Falsi", None, 0, evaluations, float("nan"), history, False)

for i in range(1, max_iter + 1):
c = (a * fb - b * fa) / (fb - fa)
```

```
fc = f(P, M, n, c)
evaluations += 1
history.append((i, c, fc))

if abs(fc) < tol or (c_old is not None and abs(c - c_old) < tol):
return result("Regula Falsi", c, i, evaluations, fc, history, True)

if fa * fc < 0:
b = c
fb = fc
else:
a = c
fa = fc

c_old = c

return result("Regula Falsi", c, max_iter, evaluations, fc, history, False)

def secant(P, M, n, x0=A, x1=B, tol=TOL, max_iter=100):
f0 = f(P, M, n, x0)
f1 = f(P, M, n, x1)
evaluations = 2
history = []

x2 = x1
f2 = f1

for i in range(1, max_iter + 1):
if abs(f1 - f0) < 1e-30:
break

x2 = x1 - f1 * (x1 - x0) / (f1 - f0)
f2 = f(P, M, n, x2)
evaluations += 1
history.append((i, x2, f2))

if abs(f2) < tol or abs(x2 - x1) < tol:
return result("Secant", x2, i, evaluations, f2, history, True)

x0, f0 = x1, f1
x1, f1 = x2, f2

return result("Secant", x2, max_iter, evaluations, f2, history, False)
```

```
def newton_raphson(P, M, n, x0=0.02, tol=TOL, max_iter=100):
x = x0
evaluations = 0
history = []

for i in range(1, max_iter + 1):
fx = f(P, M, n, x)
dfx = derivative(P, M, n, x)
evaluations += 2

if abs(dfx) < 1e-30:
break

x_new = x - fx / dfx
fx_new = f(P, M, n, x_new)
evaluations += 1
history.append((i, x_new, fx_new))

if abs(fx_new) < tol or abs(x_new - x) < tol:
return result("Newton-Raphson", x_new, i, evaluations, fx_new, history, True)

x = x_new

return result("Newton-Raphson", x, max_iter, evaluations, f(P, M, n, x), history, False)

def reference_root(P, M, n):
return bisection(P, M, n, tol=1e-15, max_iter=600)["root"]

def average_time(method, P, M, n, repeats=REPEATS):
start = perf_counter()
checksum = 0.0

for _ in range(repeats):
r = method(P, M, n)
checksum += r["root"]

end = perf_counter()

if checksum == -1:
print(checksum)

return (end - start) * 1_000_000 / repeats
```

```
methods = [bisection, regula_falsi, secant, newton_raphson]
all_results = []

for case, use, P, n, M in data:
ref = reference_root(P, M, n)

for method in methods:
r = method(P, M, n)
root = r["root"]

r["case"] = case
r["root_error"] = abs(root - ref)
r["monthly_rate_percent"] = root * 100
r["annual_rate_percent"] = ((1 + root) ** 12 - 1) * 100
r["time_microseconds"] = average_time(method, P, M, n)

all_results.append(r)

print("\nTABLE 1: ESTIMATED INTEREST RATES")
print(f"{'Case':<6}{'Root r':>15}{'Monthly %':>15}{'Annual %':>15}")

for case, use, P, n, M in data:
ref = reference_root(P, M, n)
print(
f"{case:<6}"
f"{ref:>15.10f}"
f"{ref * 100:>15.10f}"
f"{(((1 + ref) ** 12 - 1) * 100):>15.10f}"
)

print("\nTABLE 2: ITERATION COMPARISON")
print(f"{'Case':<6}{'Bisection':>12}{'Regula Falsi':>16}{'Secant':>10}{'Newton':>10}")

for case, use, P, n, M in data:
rows = [r for r in all_results if r["case"] == case]
values = {r["method"]: r["iterations"] for r in rows}

print(
f"{case:<6}"
f"{values['Bisection']:>12}"
f"{values['Regula Falsi']:>16}"
f"{values['Secant']:>10}"
f"{values['Newton-Raphson']:>10}"
```

```
)

print("\nTABLE 3: SPEED AND ACCURACY SUMMARY")
print(
f"{'Method':<17}"
f"{'Avg Iter':>12}"
f"{'Max Iter':>12}"
f"{'Avg Eval':>12}"
f"{'Mean Residual':>18}"
f"{'Max Residual':>18}"
f"{'Mean Root Err':>18}"
f"{'Max Root Err':>18}"
f"{'Avg Time us':>15}"
)

for method_name in ["Bisection", "Regula Falsi", "Secant", "Newton-Raphson"]:
rows = [r for r in all_results if r["method"] == method_name]

print(
f"{method_name:<17}"
f"{mean(r['iterations'] for r in rows):>12.2f}"
f"{max(r['iterations'] for r in rows):>12}"
f"{mean(r['evaluations'] for r in rows):>12.2f}"
f"{mean(r['residual'] for r in rows):>18.6e}"
f"{max(r['residual'] for r in rows):>18.6e}"
f"{mean(r['root_error'] for r in rows):>18.6e}"
f"{max(r['root_error'] for r in rows):>18.6e}"
f"{mean(r['time_microseconds'] for r in rows):>15.6f}"
)

print("\nTABLE 4: SAMPLE ITERATION HISTORY FOR CASE C8")
case_c8 = [item for item in data if item[0] == "C8"][0]
case, use, P, n, M = case_c8

for method in methods:
r = method(P, M, n)
print("\n", r["method"])
print(f"{'Iteration':>10}{'Approx r':>18}{'f(r)':>18}")

history = r["history"]
for it, approx, fr in history:
if it <= 5 or it % 5 == 0 or it == history[-1][0]:
print(f"{it:>10}{approx:>18.10f}{fr:>18.6e}")
```

9. Results and Analysis

The python program was compiled in a python compiler named Programiz and gave the desired results. Timing may vary depending on computer, Python version, and background processes.

9.1 Estimated Interest Rates

Table 2: Estimated monthly and effective annual rates

Case	Root r	Monthly Rate (%)	Effective Annual Rate (%)
C1	0.0184998748	1.8499874765	24.6039354694
C2	0.0168193588	1.6819358846	22.1590571132
C3	0.0220015764	2.2001576445	29.8430739033
C4	0.0175087618	1.7508761785	23.1566569400
C5	0.0190050131	1.9005013118	25.3475491631
C6	0.0144990808	1.4499080841	18.8556672980
C7	0.0169963623	1.6996362290	22.4144805072
C8	0.0155065092	1.5506509166	20.2797558133
C9	0.0182031407	1.8203140652	24.1690011189
C10	0.0194908888	1.9490888829	26.0666424014
C11	0.0205013007	2.0501300695	27.5741698767
C12	0.0130048905	1.3004890463	16.7719422982

The highest monthly interest rate was found in Case C3, where the monthly rate was approximately 2.2002% and the effective annual rate was 29.8431%. The lowest monthly interest rate was found in Case C12, where the monthly rate was approximately 1.3005% and the effective annual rate was 16.7719%.

9.2 Iteration Comparison

Table 3: Iterations required by each method

Case	Bisection	Regula Falsi	Secant	Newton-Raphson
C1	29	7	5	3
C2	29	8	5	3
C3	29	8	5	3
C4	29	9	5	3
C5	29	9	5	3
C6	29	10	5	4
C7	29	9	5	3
C8	29	10	6	4
C9	29	10	6	3
C10	29	9	5	3
C11	29	8	5	3
C12	29	10	5	4

The bisection method required 29 iterations in every case because it reduces the interval by half in each step. Regula falsi required 7 to 10 iterations. The secant method required 5 or 6 iterations. Newton-Raphson required only 3 or 4 iterations, making it the fastest in terms of iteration count.

9.3 Speed, Error, and Accuracy

Table 4: Speed and accuracy summary

Method	Average Iterations	Maximum Iterations	Average Function Evaluations	Mean Residual Error	Maximum Residual Error	Mean Root Error	Maximum Root Error	Average Speed, Microseconds
Bisection	29.00	29	31.00	2.426987e-06	5.934268e-06	5.31194e-11	9.094876e-11	9.548718
Regula Falsi	8.92	10	10.92	2.025832e-07	9.463529e-07	3.72541e-12	9.846408e-12	3.326793
Secant	5.17	6	7.17	2.921752e-11	7.639755e-11	6.54133e-16	1.649722e-15	2.102877
Newton-Raphson	3.25	4	9.75	1.421085e-11	3.365130e-11	5.44249e-16	1.370432e-15	2.618746

The speed values are representative values from one Python run using repeated execution. Actual timing may change depending on the computer, processor, Python version, and background processes. Therefore, iteration count, residual error, and root error are more stable scientific comparison measures.

10. Sample Iteration History for Case C8

For Case C8:

$$P = 98000,\ n = 30,\ M = 4110 \tag{15}$$

The nonlinear equation is:

$$f(r) = \frac{98000r(1+r)^{30}}{(1+r)^{30}-1} - 4110 = 0 \tag{16}$$

10.1 Bisection Method

Table 5: Iterations by bisection method

Iteration	Approximation r	f(r)
1	0.0250000000	5.722088e+02
2	0.0125000000	-1.725027e+02
3	0.0187500000	1.908436e+02

4	0.0156250000	6.885609e+00
5	0.0140625000	-8.338348e+01
10	0.0154785156	-1.625770e+00
15	0.0155075073	5.797643e-02
20	0.0155065060	-1.858482e-04
25	0.0155065104	7.380376e-05
29	0.0155065092	3.481338e-06

The bisection method converged to the desired solution in 29 iterations.

10.2 Regula Falsi Method

Table 6: Iterations by regula falsi method

Iteration	**Approximation** r	f(r)
1	0.0135655063	-1.118552e+02
2	0.0152800906	-1.313910e+01
3	0.0154803331	-1.520229e+00
4	0.0155034861	-1.755848e-01
5	0.0155061601	-2.027573e-02
10	0.0155065092	-4.162839e-07

The regula falsi method converged to the desired solution in 10 iterations.

10.3 Secant Method

Table 7: Iterations by secant method

Iteration	**Approximation** r	f(r)
1	0.0135655063	-1.118552e+02
2	0.0152800906	-1.313910e+01
3	0.0155083015	1.041067e-01
4	0.0155065075	-9.512454e-05
5	0.0155065092	-6.912160e-10
6	0.0155065092	2.728484e-11

The secant method converged to the desired solution in 6 iterations.

10.4 Newton-Raphson Method

Table 8: Iterations by Newton-Raphson method

Iteration	**Approximation** r	f(r)
1	0.0155840310	4.504120e+00
2	0.0155065334	1.405751e-03
3	0.0155065092	1.327862e-10
4	0.0155065092	-1.364242e-11

The Newton-Raphson method converged to the desired solution in 4 iterations.

11. Discussion

The results show that all four numerical methods successfully solved the nonlinear EMI equation. This proves that numerical methods can be applied to real-life financial problems where direct algebraic solutions are difficult.

The bisection method was the slowest method, requiring 29 iterations in all cases. However, it is reliable because it keeps the root inside a fixed interval. This makes it useful for beginners and for cases where guaranteed convergence is important.

The regula falsi method performed better than bisection. It required fewer iterations because it uses the function values at the endpoints to estimate the root. However, it was still slower than the secant and Newton-Raphson methods in this study.

The secant method performed very well. It required only 5 or 6 iterations and gave very small residual and root errors. An important advantage of the secant method is that it does not require the derivative of the function.

The Newton-Raphson method required the fewest iterations, with an average of 3.25 iterations. It also gave the smallest mean residual error. However, it requires the derivative and depends on a suitable initial guess. Therefore, it is very effective for programmed computation but may be less convenient for manual calculation.

The effective annual rate is more useful than simply comparing the extra amount paid. A consumer may see a monthly EMI as affordable, but the true cost depends on the principal, EMI, and tenure together. Numerical methods can reveal this hidden interest rate and support better financial decision-making.

12. Major Findings

The major findings of the study are:

1. The household instalment problem leads to a nonlinear equation when the interest rate is unknown.
2. All four numerical methods successfully solved the EMI equation.
3. Newton-Raphson was the fastest method in terms of iteration count.
4. The secant method gave excellent accuracy without using a derivative.
5. Regula falsi was faster than bisection but slower than secant and Newton-Raphson.
6. Bisection was the slowest but most reliable bracketing method.
7. The highest effective annual rate was 29.8431% in Case C3.
8. The lowest effective annual rate was 16.7719% in Case C12.
9. Numerical methods can help consumers estimate hidden borrowing costs in household instalment plans.

13. Limitations of the Study

This study has some limitations. The study assumes fixed monthly instalments and does not include processing fees, late payment charges, insurance charges, taxes, or service fees. The execution time may vary depending on the computer system and Python environment. Newton-Raphson performance depends on the chosen initial guess.

14. Future Scope

Future research can include processing fees and other charges to estimate a more realistic annual percentage rate. Further studies may compare additional methods such as fixed-point iteration, modified

Newton method, hybrid bisection-secant method, and Brent's method.

15. Conclusion

This paper presented a comparative study of numerical methods for estimating effective interest rates in household instalment plans. The real-life problem was converted into a nonlinear EMI equation. Four numerical methods, namely bisection, regula falsi, secant, and Newton-Raphson methods, were solved using Python programming. The dataset of twelve realistic household instalment cases were used.

The results showed that Newton-Raphson was the fastest method in terms of iteration count, while the secant method gave excellent accuracy without requiring derivative calculation. Regula falsi performed better than bisection, while bisection remained the simplest and most dependable method. The study proves that numerical methods are useful not only in theoretical mathematics but also in real-life financial decision-making.

References

1. Ahmad, A. G., "Comparative Study of Bisection and Newton-Rhapson Methods of Root-Finding Problems", International Journal of Mathematics Trends and Technology, (2015), 19(2), 121–129. Seventh Sense Research Group, India. DOI: 10.14445/22315373/IJMTT-V19P516.
2. Azure, I., Aloliga, G., & Doabil, L, "Comparative Study of Numerical Methods for Solving Non-linear Equations Using Manual Computation", Mathematics Letters, (2020), 5(4), 41–46. Science Publishing Group, online journal. DOI: 10.11648/j.ml.20190504.11.
3. Badr, E., Almotairi, S., & El Ghamry, A., "A Comparative Study among New Hybrid Root Finding Algorithms and Traditional Methods". Mathematics, (2021) 9(11), 1306. Basel, Switzerland: MDPI. DOI: 10.3390/math9111306.
4. Flórez, M., Vera, M., Salazar-Torres, J., Huérfano, Y., Gelvez-Almeida, E., Valbuena, O., Vera, M. I., & Aranguen, M., "Interest Rates Calculation in Certain Ordinary Annuities", Journal of Physics: Conference Series,(2019), 1414, 012009. Place/Publisher: Bristol, United Kingdom: IOP Publishing. DOI: 10.1088/1742-6596/1414/1/012009.
5. Tampubolon, M. N., Ditasona, C., Siregar, J., & Simbolon, R., "Application of the Newton-Raphson Method to Compare Effective Interest Rates for Motorcycle Installments", EduMatSains: Jurnal Pendidikan, Matematika dan Sains, (2026), 10(4), 265–275. Jakarta, Indonesia: Fakultas Keguruan dan Ilmu Pendidikan, Universitas Kristen Indonesia. DOI: 10.33541/edumatsains.v10i4.7808.
6. Burden, R. L., Faires, J. D., & Burden, A. M., Numerical Analysis (10th ed.), (2016), Boston, MA, USA: Cengage Learning.
7. Chapra, S. C., & Canale, R. P., Numerical Methods for Engineers (7th ed.), (2015), New York, USA: McGraw-Hill Education.
8. Kellison, S. G., The Theory of Interest (3rd ed.), (2008), New York, USA: McGraw-Hill/Irwin.
9. Broverman, S. A., Mathematics of Investment and Credit (5th ed.), (2010), Winsted, Connecticut, USA: ACTEX Publications.