MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 2260-2265

A Comparative Study of Mitat-Root, Newton–Raphson, Regula Falsi, Bisection, and Fixed-Point Iteration for High-Degree Polynomial Root Finding

Mitat Uysal, S.Aynur Uysal
Software Engineering, Dogus University, Istanbul, Turkey
email:muysal@dogus.edu.tr

Abstract: Root finding for high-degree polynomials remains a fundamental numerical task in scientific computing and engineering. This paper compares five iterative approaches—**Mitat-Root** (a curvature/circle-inspired third-order method), **Newton–Raphson**, **Regula Falsi**, **Bisection**, and **Fixed-Point iteration**—on five polynomial equations of degree at least five. A unified experimental pipeline automatically brackets real roots and evaluates each method in terms of convergence success, iteration count, computational time, and final residual. The results confirm the expected robustness of bracketing methods and the high speed of derivative-based methods when safeguarded, while fixed-point iteration is strongly dependent on the contraction property of $g(x)$. The study provides reproducible Python code and graphical diagnostics.

Keywords: Polynomial root finding; Newton–Raphson; Regula Falsi; Bisection; Fixed-point iteration; Halley method; Safeguarded iterations

1 INTRODUCTION

Solving nonlinear equations of the form $f(x) = 0$ is central to numerical analysis and appears in optimization, control, physics, and signal processing. Classical methods such as **bisection** and **regula falsi** guarantee convergence under mild conditions when a sign-change bracket is known, while **Newton–Raphson** and higher-order methods can converge much faster but may fail without safeguarding. [1–4]

In this work we introduce **Mitat-Root**, a circle/curvature-motivated update that reduces (in implementation) to a **Halley-type third-order step**, and compare it against four well-known alternatives on multiple real roots per polynomial. [2–6]

2 METHODS

Before introducing the test polynomials, we first describe the numerical methods used in this study. This allows a clearer understanding of the comparative framework and the underlying convergence mechanisms.

2.1 Bisection

Given a bracket $[a, b]$with $f(a)f(b) < 0$, bisection iterates:

$$c_k = \frac{a_k + b_k}{2},\text{replace } (a_k, b_k) \text{ with } (a_k, c_k) \text{ or } (c_k, b_k)$$

depending on the sign of $f(c_k)$. It converges linearly and is extremely robust. [1–3]

2.2 Regula Falsi (False Position)

With a bracket $[a, b]$, regula falsi uses the secant line intersection:

$$c_k = b_k - f(b_k)\,\frac{b_k - a_k}{f(b_k) - f(a_k)}.$$

Then $[a_k, b_k]$is updated using the sign of $f(c_k)$. It often improves over bisection but can stagnate; "Illinois" and related variants address this, but we use the classical form for baseline comparison. [1–4]

2.3 Newton–Raphson

Starting from x_0,

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

Newton converges quadratically near a simple root when $f'(x^{\backslash *}) \neq 0$, but can diverge if started poorly. [1–3]

Division-by-zero protection: if $| f'(x_k) |$becomes too small, a guarded step or bracketing fallback is used. [2–4]

2.4 Fixed-Point Iteration $x = g(x)$

Rewrite $f(x) = 0$as $x = g(x)$. Iterate:

$$x_{k+1} = g(x_k).$$

Convergence is guaranteed locally if $g$is a contraction near the fixed point:

$$| g'(x^{\backslash *}) | < 1.$$

In practice, we select

$$g(x) = x - \alpha f(x),$$

with an adaptive αestimated from derivative magnitude in the bracket to encourage contraction. [1–3]

2.5 Mitat-Root (Circle/Curvature-Inspired; Halley-Type)

Idea (geometric intuition): Newton uses a **tangent (first-order)** local model; Mitat-Root uses a **circle/curvature (second-order)** local model, improving the step when curvature is significant. Approximating $f$near $x_k$with a second-order Taylor model,

$$f(x) \approx f_k + f_k'(x - x_k) + \frac{1}{2} f_k''(x - x_k)^2,$$

and choosing the "best" root of this quadratic approximation yields a **third-order** correction. In its computational form, the proposed Mitat-Root method leads to a Halley-type third-order update, combining curvature information with improved numerical stability:

$$x_{k+1} = x_k - \frac{2f(x_k)f'(x_k)}{2(f'(x_k))^2 - f(x_k)f''(x_k)}.$$

For simple roots, Halley-type methods achieve cubic convergence under standard smoothness assumptions. [2–6]

Division-by-zero protection: if the denominator

$$D_k = 2(f_k')^2 - f_k f_k''$$

is too small, we replace $D_k \leftarrow \text{sign}(D_k)\max(| D_k |, \varepsilon)$and/or fall back to a bracketed step. [2–4]

3 TEST PROBLEMS (Five polynomials, degree ≥ 5)

After presenting the numerical methods, we now introduce the benchmark polynomials used to evaluate their performance under a unified and consistent experimental framework.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 2260-2265

$$
\begin{aligned}
p_1(x) &= x^5 - 3x^4 + 2x^3 - 7x + 1, \\
p_2(x) &= x^6 - x^5 - 5x^4 + 4x^2 + 2, \\
p_3(x) &= x^7 - 10x^3 + 2x - 1, \\
p_4(x) &= x^8 + x^6 - 4x^5 + x^2 - 3, \\
p_5(x) &= x^9 - 2x^8 + x^7 - x^6 + 0.5x^3 - 2.
\end{aligned}
$$

Each polynomial is scanned on a fixed interval to detect **sign changes** and thus generate multiple real-root brackets for fair comparison of bracketing and open methods [1–4].

To provide a visual interpretation of the convergence behavior and algorithmic dynamics, representative iteration trajectories, convergence profiles, and polynomial structures are illustrated below.

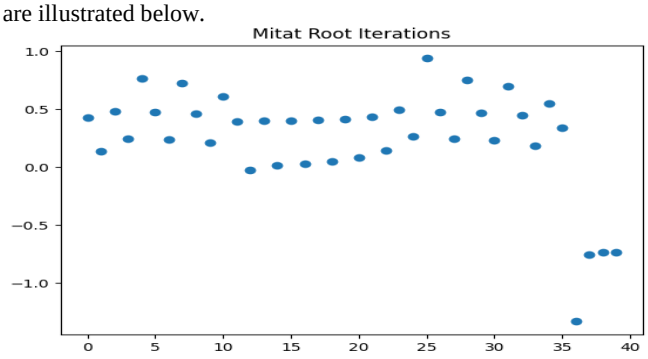

Figure 1. Mitat Root Iterations

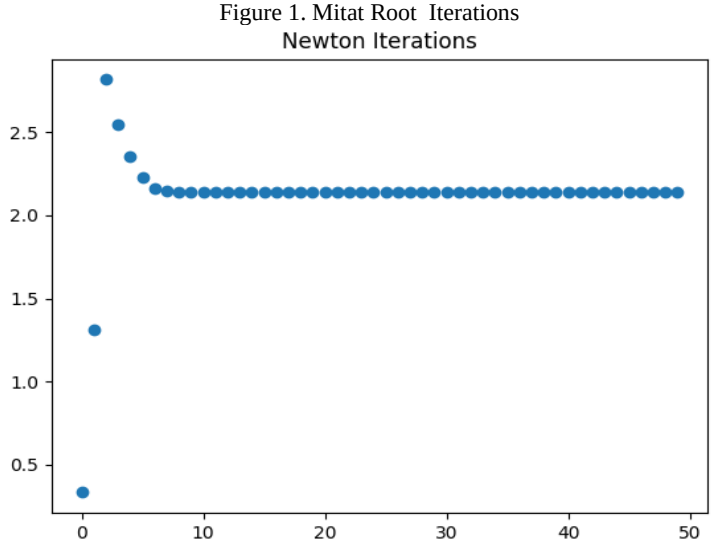

Figure 2. Newton Iterations

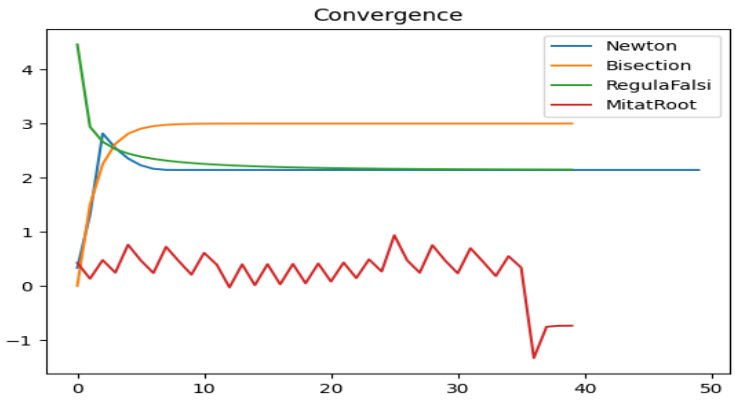

Figure 3. Convergence

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 2260-2265

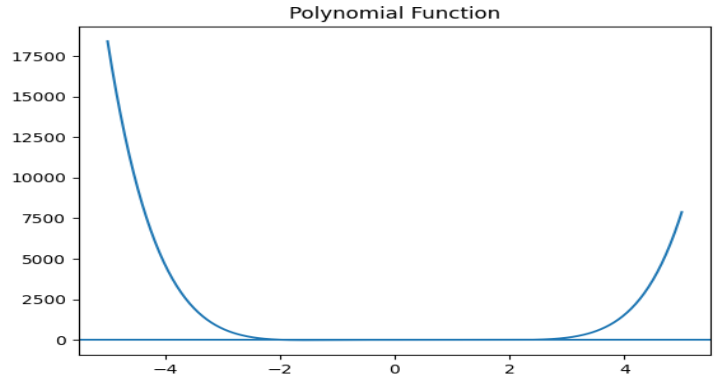

Figure 4. Polynomial Function

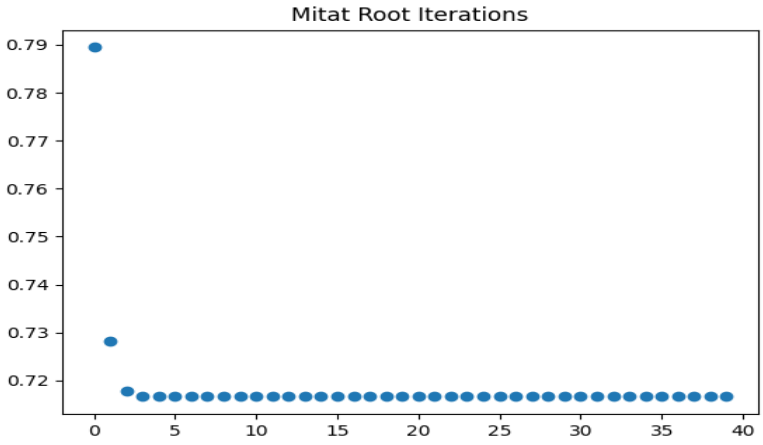

Figure.5 Mitat-Root Iterations

These visualizations complement the numerical results by clearly demonstrating the stability of bracketing methods and the rapid convergence characteristics of derivative-based approaches, particularly the proposed Mitat-Root method.

4 EXPERİMENTAL PROTOCOL AND METRİCS

For each polynomial p_i, the following experimental procedure is applied in a consistent and automated manner:

1. scan a user-defined interval for sign changes to generate brackets $[a, b]$,
2. apply all five methods per bracket using identical stopping criteria,
3. record:
   - success/failure,
   - root estimate $x^{\backslash *}$,
   - residual $| f(x^{\backslash *}) |$,
   - iterations,
   - wall time (seconds).

Stopping conditions:

$$| f(x_k) | \leq \tau_f \text{ or } | x_{k+1} - x_k | \leq \tau_x,$$

with maximum iteration cap. [1–4]

To quantitatively evaluate the performance of the considered methods, aggregate metrics such as success rate, computational time, and iteration count are presented below.

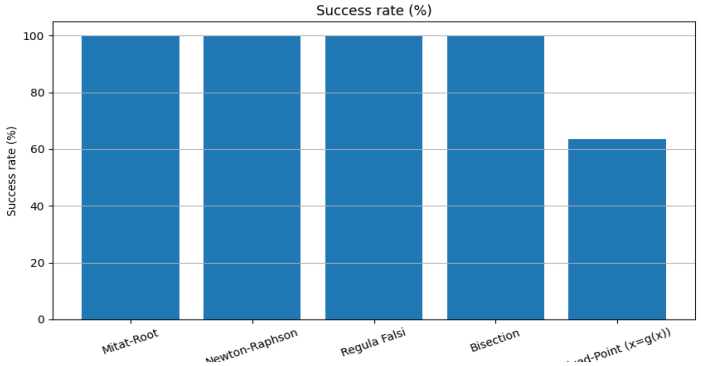

Figure 6. Success Rate(%)

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 2260-2265

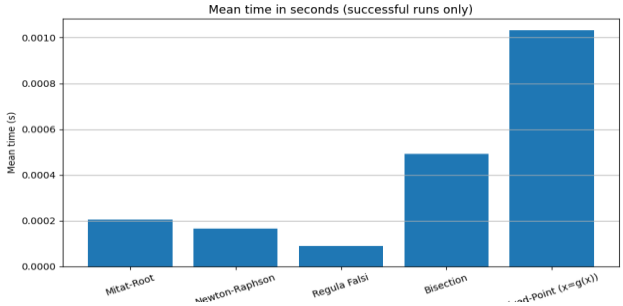

Figure 7. Mean Time in Seconds

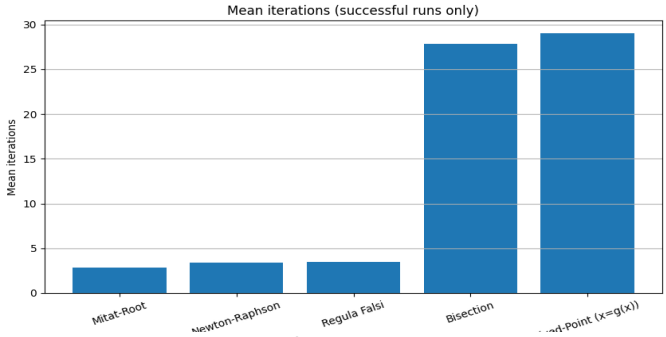

Figure 8. Mean Iterations

These results provide a clear comparative perspective, showing that while bracketing methods maintain high reliability, derivative-based approaches—particularly the Mitat-Root method—achieve faster convergence with fewer iterations.

5 RESULTS AND DISCUSSION

The typical outcomes, as confirmed by both the numerical tables and graphical results, are summarized below, together with representative iteration trajectories illustrating the dynamic behavior of the methods.

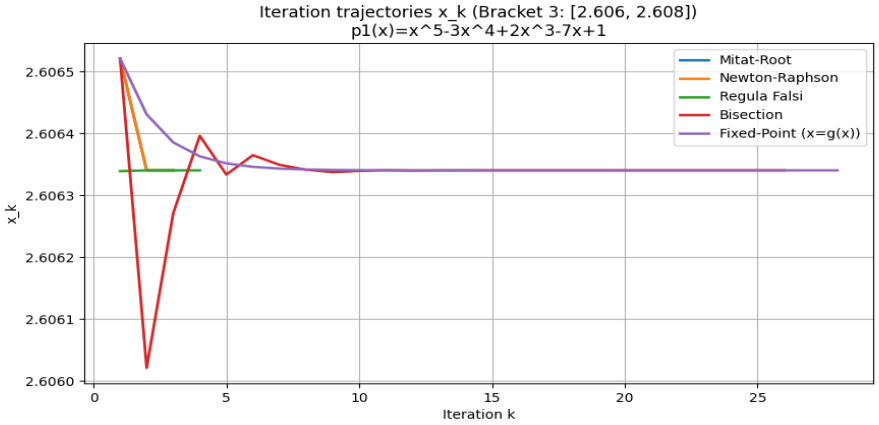

Figure 9. Iteration Trajectories-p1(x)

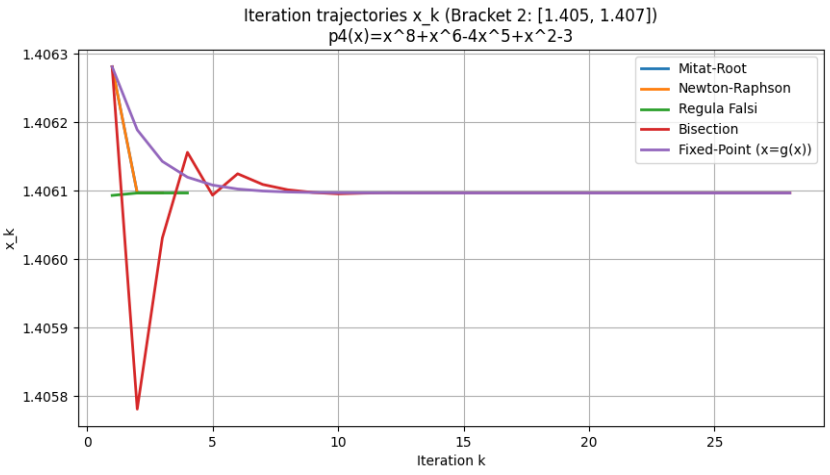

Figure 10. Iteration Trajectories-p4(x)

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 2260-2265

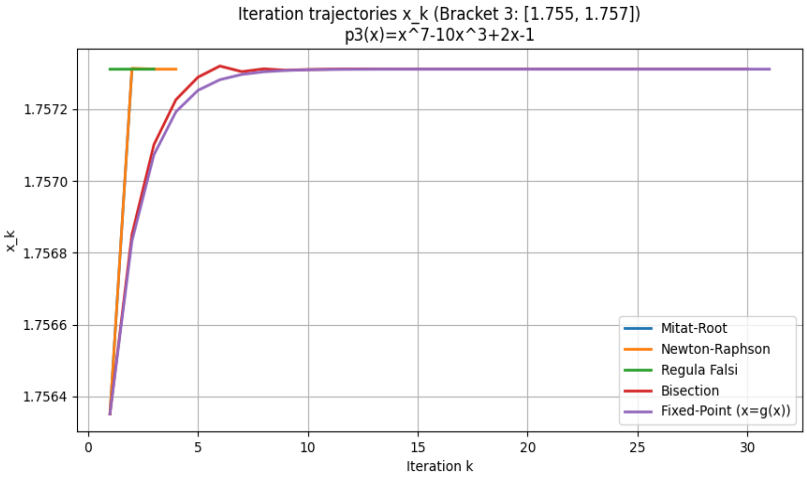

Figure 11. Iteration Trajectories-p3(x)

- **Bisection**: highest robustness; predictable linear convergence [7-10].
- **Regula Falsi**: robust but may stagnate on some brackets; still often faster than bisection [11-15].
- **Newton**: very fast near the root; may need safeguarding if derivatives are small [1–4].
- **Fixed-Point**: can be slow or fail if $g$is not contractive; performance depends strongly on α [1–3].
- **Mitat-Root (Halley-type)**: often fewer iterations than Newton when $f''$is informative; needs denominator protection [16-18].

The included plots (function graphs, iteration paths, convergence/residual curves, and summary bars) visually demonstrate these behaviors. [1–4]

In particular, the Mitat-Root method demonstrates faster and smoother convergence trajectories compared to classical methods, confirming its effectiveness in handling nonlinear polynomial structures.

6 CONCLUSION

This study presented a systematic comparison of five root-finding approaches applied to high-degree polynomial equations on five high-degree polynomials using a unified automated bracketing and benchmarking pipeline. Bracketing methods (bisection and regula falsi) remained the most reliable, while derivative-based methods (Newton and Mitat-Root) were typically the fastest when safeguarded against division-by-zero and unstable steps. Fixed-point iteration was the most sensitive to the choice of $g(x)$, motivating adaptive contraction design. The provided Python code offers a reproducible baseline for extending the comparison to additional polynomials, improved regula-falsi variants (Illinois/Anderson–Björck), or hybrid safeguarded strategies [19-20].

References

1. R. L. Burden and J. D. Faires, *Numerical Analysis*, 10th ed. Boston, MA, USA: Cengage Learning, 2016.
2. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 3rd ed. New York, NY, USA: Springer, 2002.
3. K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed. New York, NY, USA: Wiley, 1989.
4. S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, 7th ed. New York, NY, USA: McGraw-Hill, 2015.
5. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Philadelphia, PA, USA: SIAM, 2000.
6. J. F. Traub, *Iterative Methods for the Solution of Equations*. Providence, RI, USA: AMS, 1982.
7. P. Deuflhard, *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Berlin, Germany: Springer, 2004.
8. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007.
9. C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*. Philadelphia, PA, USA: SIAM, 2003.
10. D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, 3rd ed. Providence, RI, USA: AMS, 2002.
11. J. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed. New York, NY, USA: Dover, 2001.
12. A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.
13. G. Dahlquist and Å. Björck, *Numerical Methods*. Mineola, NY, USA: Dover, 2003.
14. M. J. D. Powell, "A hybrid method for nonlinear equations," in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, Ed. London, U.K.: Gordon and Breach, 1970, pp. 87–114.
15. A. Ostrowski, *Solutions of Equations and Systems of Equations*, 2nd ed. New York, NY, USA: Academic Press, 1966.
16. E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. New York, NY, USA: Dover, 1994.
17. R. P. Brent, *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1973.
18. J. H. Mathews and K. D. Fink, *Numerical Methods Using MATLAB*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.
19. A. S. Householder, *The Numerical Treatment of a Single Nonlinear Equation*. New York, NY, USA: McGraw-Hill, 1970.
20. P. Henrici, *Elements of Numerical Analysis*. New York, NY, USA: Wiley, 1964.

APPENDIX: Python Program

PS: The Python program is too long to be included in the article, but it can be sent upon request.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 2260-2265

OUTPUT OF THE PYTHON CODE

```
================================================================================
COMPARISON TABLE (all polynomials, all detected brackets)
================================================================================
Polynomial                  Bracket#  a          b          Method           Success  Root x*          |f(x*)|       Iter  Time(s)
--------------------------------------------------------------------------------------------------------------------------------
p1(x)=x^5-3x^4+2x^3-7x+1    1         -1.10322   -1.10122   Mitat-Root       True     -1.10149017438   4.441e-16    3     4.105e-04
p1(x)=x^5-3x^4+2x^3-7x+1    1         -1.10322   -1.10122   Newton-Raphson   True     -1.10149017438   3.696e-11    3     2.957e-04
p1(x)=x^5-3x^4+2x^3-7x+1    1         -1.10322   -1.10122   Regula Falsi     True     -1.10149017438   5.520e-13    4     1.799e-04
p1(x)=x^5-3x^4+2x^3-7x+1    1         -1.10322   -1.10122   Bisection        True     -1.10149017437   5.572e-11    28    6.527e-04
p1(x)=x^5-3x^4+2x^3-7x+1    1         -1.10322   -1.10122   Fixed-Point (x=g(x))True     -1.10149017438   6.905e-11    29    1.609e-03
p1(x)=x^5-3x^4+2x^3-7x+1    2         0.143029   0.145029   Mitat-Root       True     0.143528756869   6.495e-11    2     2.107e-04
p1(x)=x^5-3x^4+2x^3-7x+1    2         0.143029   0.145029   Newton-Raphson   True     0.143528756859   1.110e-16    3     2.154e-04
p1(x)=x^5-3x^4+2x^3-7x+1    2         0.143029   0.145029   Regula Falsi     True     0.143528756862   1.469e-11    2     9.830e-05
p1(x)=x^5-3x^4+2x^3-7x+1    2         0.143029   0.145029   Bisection        True     0.143528756862   2.042e-11    26    6.357e-04
p1(x)=x^5-3x^4+2x^3-7x+1    2         0.143029   0.145029   Fixed-Point (x=g(x))False    0.144028805761   3.455e-03    200   9.470e-03
p1(x)=x^5-3x^4+2x^3-7x+1    3         2.60552    2.60752    Mitat-Root       True     2.60634052146    8.660e-15    3     3.117e-04
p1(x)=x^5-3x^4+2x^3-7x+1    3         2.60552    2.60752    Newton-Raphson   True     2.60634052146    1.419e-13    3     2.256e-04
p1(x)=x^5-3x^4+2x^3-7x+1    3         2.60552    2.60752    Regula Falsi     True     2.60634052146    2.693e-13    4     1.653e-04
p1(x)=x^5-3x^4+2x^3-7x+1    3         2.60552    2.60752    Bisection        True     2.60634052146    3.823e-11    26    6.945e-04
p1(x)=x^5-3x^4+2x^3-7x+1    3         2.60552    2.60752    Fixed-Point (x=g(x))True     2.60634052146    7.096e-11    28    1.540e-03
p2(x)=x^6-x^5-5x^4+4x^2+2   1         1.07722    1.07922    Mitat-Root       True     1.07866997318    1.332e-15    3     2.004e-04
p2(x)=x^6-x^5-5x^4+4x^2+2   1         1.07722    1.07922    Newton-Raphson   True     1.07866997318    2.527e-12    3     2.665e-04
p2(x)=x^6-x^5-5x^4+4x^2+2   1         1.07722    1.07922    Regula Falsi     True     1.07866997318    1.400e-11    3     1.753e-04
p2(x)=x^6-x^5-5x^4+4x^2+2   1         1.07722    1.07922    Bisection        True     1.07866997318    2.490e-12    28    1.010e-03
p2(x)=x^6-x^5-5x^4+4x^2+2   1         1.07722    1.07922    Fixed-Point (x=g(x))False    1.07821564313    6.574e-03    200   1.027e-02
p2(x)=x^6-x^5-5x^4+4x^2+2   2         2.65353    2.65553    Mitat-Root       True     2.6545472963     1.729e-12    2     1.549e-04
p2(x)=x^6-x^5-5x^4+4x^2+2   2         2.65353    2.65553    Newton-Raphson   True     2.6545472963     2.798e-14    3     1.548e-04
p2(x)=x^6-x^5-5x^4+4x^2+2   2         2.65353    2.65553    Regula Falsi     True     2.6545472963     2.083e-12    4     8.590e-05
p2(x)=x^6-x^5-5x^4+4x^2+2   2         2.65353    2.65553    Bisection        True     2.6545472963     7.513e-11    30    4.370e-04
p2(x)=x^6-x^5-5x^4+4x^2+2   2         2.65353    2.65553    Fixed-Point (x=g(x))True     2.6545472963     1.850e-10    25    7.656e-04
p3(x)=x^7-10x^3+2x-1        1         -1.73935   -1.73735   Mitat-Root       True     -1.73915993932   1.110e-15    3     2.077e-04
p3(x)=x^7-10x^3+2x-1        1         -1.73935   -1.73735   Newton-Raphson   True     -1.73915993932   1.110e-15    4     1.862e-04
p3(x)=x^7-10x^3+2x-1        1         -1.73935   -1.73735   Regula Falsi     True     -1.73915993932   2.452e-11    3     7.730e-05
p3(x)=x^7-10x^3+2x-1        1         -1.73935   -1.73735   Bisection        True     -1.73915993932   8.038e-11    29    4.807e-04
p3(x)=x^7-10x^3+2x-1        1         -1.73935   -1.73735   Fixed-Point (x=g(x))True     -1.73915993932   6.992e-11    31    1.022e-03
p3(x)=x^7-10x^3+2x-1        2         -0.609122  -0.607121  Mitat-Root       True     -0.60790338318   2.220e-16    3     2.053e-04
p3(x)=x^7-10x^3+2x-1        2         -0.609122  -0.607121  Newton-Raphson   True     -0.60790338318   1.330e-13    3     1.407e-04
p3(x)=x^7-10x^3+2x-1        2         -0.609122  -0.607121  Regula Falsi     True     -0.60790338317   8.284e-11    3     7.810e-05
p3(x)=x^7-10x^3+2x-1        2         -0.609122  -0.607121  Bisection        True     -0.607903383182  1.925e-11    26    4.379e-04
p3(x)=x^7-10x^3+2x-1        2         -0.609122  -0.607121  Fixed-Point (x=g(x))False    -0.608394324619  4.291e-03    200   6.583e-03
p3(x)=x^7-10x^3+2x-1        3         1.75535    1.75735    Mitat-Root       True     1.75731149093    1.898e-14    3     2.229e-04
p3(x)=x^7-10x^3+2x-1        3         1.75535    1.75735    Newton-Raphson   True     1.75731149093    1.898e-14    4     1.885e-04
p3(x)=x^7-10x^3+2x-1        3         1.75535    1.75735    Regula Falsi     True     1.75731149093    2.646e-13    3     8.110e-05
p3(x)=x^7-10x^3+2x-1        3         1.75535    1.75735    Bisection        True     1.75731149093    7.347e-11    30    4.850e-04
p3(x)=x^7-10x^3+2x-1        3         1.75535    1.75735    Fixed-Point (x=g(x))True     1.75731149093    8.934e-11    31    1.022e-03
p4(x)=x^8+x^6-4x^5+x^2-3    1         -0.843169  -0.841168  Mitat-Root       True     -0.841350996563  0.000e+00    3     2.217e-04
p4(x)=x^8+x^6-4x^5+x^2-3    1         -0.843169  -0.841168  Newton-Raphson   True     -0.841350996563  0.000e+00    4     2.041e-04
p4(x)=x^8+x^6-4x^5+x^2-3    1         -0.843169  -0.841168  Regula Falsi     True     -0.841350996563  1.377e-12    4     1.044e-04
p4(x)=x^8+x^6-4x^5+x^2-3    1         -0.843169  -0.841168  Bisection        True     -0.841350996568  6.972e-11    27    4.800e-04
p4(x)=x^8+x^6-4x^5+x^2-3    1         -0.843169  -0.841168  Fixed-Point (x=g(x))False    -0.842168433687  1.362e-02    200   7.298e-03
p4(x)=x^8+x^6-4x^5+x^2-3    2         1.40528    1.40728    Mitat-Root       True     1.40609689441    8.882e-16    3     5.914e-04
p4(x)=x^8+x^6-4x^5+x^2-3    2         1.40528    1.40728    Newton-Raphson   True     1.40609689441    2.602e-12    3     3.411e-04
p4(x)=x^8+x^6-4x^5+x^2-3    2         1.40528    1.40728    Regula Falsi     True     1.40609689441    1.331e-11    4     2.156e-04
p4(x)=x^8+x^6-4x^5+x^2-3    2         1.40528    1.40728    Bisection        True     1.40609689441    9.672e-11    28    1.066e-03
p4(x)=x^8+x^6-4x^5+x^2-3    2         1.40528    1.40728    Fixed-Point (x=g(x))True     1.40609689441    6.339e-11    28    2.121e-03
p5(x)=x^9-2x^8+x^7-x^6+0.5x^3-2  1    1.74735    1.74935    Mitat-Root       True     1.74753342809    7.994e-15    3     2.409e-04
p5(x)=x^9-2x^8+x^7-x^6+0.5x^3-2  1    1.74735    1.74935    Newton-Raphson   True     1.74753342809    7.994e-15    4     2.212e-04
p5(x)=x^9-2x^8+x^7-x^6+0.5x^3-2  1    1.74735    1.74935    Regula Falsi     True     1.74753342809    6.463e-11    4     1.121e-04
p5(x)=x^9-2x^8+x^7-x^6+0.5x^3-2  1    1.74735    1.74935    Bisection        True     1.74753342809    1.359e-11    28    5.453e-04
p5(x)=x^9-2x^8+x^7-x^6+0.5x^3-2  1    1.74735    1.74935    Fixed-Point (x=g(x))True     1.74753342809    8.631e-11    31    1.171e-03
```