M 348 HOMEWORK 3

ZUN CAO

1. Chapter 1.4 Ex # 4 Estimate e_{i+1} as in Exercise 3.

$$(a) 32x^3 - 32x^2 - 6x + 9 = 0; r = -\frac{1}{2}, r = \frac{3}{4} \quad (b) x^3 - x^2 - 5x - 3 = 0; r = -1, r = 3$$

Solution. (a) Let $f(x) = 32x^3 - 32x^2 - 6x + 9$. Differentiate $f(x)$, we get

$$f'(x) = 96x^2 - 64x - 6$$

$$f''(x) = 192x - 64$$

At $r = -\frac{1}{2}$, $\quad f'(-\frac{1}{2}) = 50, \quad f''(-\frac{1}{2}) = -160.$
At $r = \frac{3}{4}$, $\quad f'(\frac{3}{4}) = 0, \quad f''(\frac{3}{4}) = 80.$
Now consider $r = -\frac{1}{2}$. Since $f'(-\frac{1}{2}) \neq 0$, we can apply Theorem 1.11(Quadratic convergent). M can be calculate by this equation:

$$\begin{aligned} M &= \frac{f''(r)}{2f'(r)} \\ &= \frac{f''(-\frac{1}{2})}{2 \cdot f'(-\frac{1}{2})} \\ &= \frac{192(-\frac{1}{2}) - 64}{2 \cdot (96(-\frac{1}{2})^2 - 64(-\frac{1}{2}) - 6)} \\ &= \frac{-160}{2 \cdot 50} \\ &= -1.6 \end{aligned}$$

Applying the theorem,

$$\begin{aligned} e_{i+1} &= |\frac{f''(r)}{2f'(r)}|e_i^2 \\ &= |-1.6|e_i^2 \\ &= 1.6e_i^2 \end{aligned}$$

Date: 2/11/2024.

Then, let's consider $r = \frac{3}{4}$. Since $f'(\frac{3}{4}) = 0$, Theorem 1.11 cannot be used. Therefore, we try to use Theorem 1.12. Note that $f''(\frac{3}{4}) = 192(\frac{3}{4}) - 64 = 80 \neq 0$. Therefore, it is $(2+1)$ times continuously differentiable function, and

$$e_{i+1} = \frac{m-1}{m} e_i = \frac{2-1}{2} e_i = \frac{1}{2} e_i$$

(b) Let $x^3 - x^2 - 5x - 3$. Differentiate $f(x)$, we get

$$f'(x) = 3x^2 - 2x - 5$$

$$f''(x) = 6x - 2$$

At $r = -1, \quad f'(-1) = 0, \quad f''(-1) = -8.$
At $r = 3, \quad f'(3) = 16, \quad f''(3) = 16.$
Now consider $r = -1$. Since $f'(-1) = 0$, we cannot apply Theorem 1.11(Quadratic convergent). So consider Theorem 1.12:

$$\begin{aligned} e_{i+1} &= \frac{m-1}{m} e_i \\ &= \frac{2-1}{2} e_i \\ &= \frac{1}{2} e_i \end{aligned}$$

Then, let's consider $r = 3$. Since $f'(3) \neq 0$, Theorem 1.11 can be used:

$$\begin{aligned} M &= \frac{f''(r)}{2f'(r)} \\ &= \frac{f''(3}{2 \cdot f'(3)} \\ &= \frac{6 \cdot 3 - 2}{2 \cdot (3 \cdot 3^2 - 2 \cdot 3 - 5)} \\ &= \frac{1}{2} \end{aligned}$$

Applying the theorem,

$$\begin{aligned} e_{i+1} &= |\frac{f''(r)}{2f'(r)}| e_i^2 \\ &= |\frac{1}{2}| e_i^2 \\ &= \frac{1}{2} e_i^2 \end{aligned}$$

2. Additional Problem 1 Find each fixed point and decide whether Fixed-Point Iteration is locally convergent to it. If convergent, find the rate.

$$(a) g_1(x) = \frac{4x}{x^2+3} \quad (b) g_2(x) = \frac{x^2-5x}{x^2+x-6}$$

Solution. (a) To find the fixed point, we let:

$$x = \frac{4x}{x^2+3}$$

Solving this equation, we get:

$$x = \frac{4x}{x^2+3}$$

$$1 = \frac{4}{x^2+3}$$

$$4 = x^2+3$$

$$x^2 = 1$$

$$x^* = 1, \quad x^* = -1, \quad x^* = 0$$

To test convergence, we need to find $g_1'(x)$ and evaluate it at the fixed points. The condition for local convergence is $|g_1'(x^*)| < 1$. Note that:

$$g_1'(x^*) = -\frac{4(x^2-3)}{(x^2+3)^2}$$

When $x^* = 1$,

$$|g_1'(x^*)| = -\frac{4(1-3)}{(1+3)^2} = \frac{1}{2}$$

When $x^* = -1$,

$$|g_1'(x^*)| = -\frac{4(1-3)}{(1+3)^2} = \frac{1}{2}$$

When $x^* = 0$,

$$|g_1'(x^*)| = -\frac{4(0-3)}{(0+3)^2} = \frac{4}{3}$$

Since when $x^* = 1$ and $x^* = -1$, $|g_1'(x^*)| = \frac{1}{2} < 1$, indicating local convergence with linear rate.

Since when $x^* = 0$, $|g_1'(x^*)| = \frac{4}{3} > 1$, indicating that Fixed-Point Iteration is not locally convergent at this point.

(b) To find the fixed point, we let:

$$x = \frac{x^2 - 5x}{x^2 + x - 6}$$

Solving this equation, we get:

$$\begin{aligned} x &= \frac{x^2 - 5x}{x^2 + x - 6} \\ 1 &= \frac{x - 5}{x^2 + x - 6} \\ x - 5 &= x^2 + x - 6 \\ x^2 &= 1 \end{aligned}$$

$$x^* = 1, \quad x^* = -1, \quad x^* = 0$$

To test convergence, we need to find $g_1'(x)$ and evaluate it at the fixed points. The condition for local convergence is $|g_1'(x^*)| < 1$. Note that:

$$g_1'(x^*) = -\frac{6(x^2 - 2x + 5)}{(x^2 + x - 6)^2}$$

When $x^* = 1$,

$$|g_1'(x^*)| = -\frac{6(1 - 2 \cdot 1 + 5)}{(1 + 1 - 6)^2} = \frac{3}{2}$$

When $x^* = -1$,

$$|g_1'(x^*)| = -\frac{6(1 + 2 \cdot 1 + 5)}{(1 - 1 - 6)^2} = \frac{4}{3}$$

When $x^* = 0$,

$$|g_1'(x^*)| = -\frac{6(0 + 5)}{(0 - 6)^2} = \frac{5}{6}$$

Since when $x^* = 1$, $|g_1'(x^*)| = \frac{3}{2} > 1$, indicating that Fixed-Point Iteration is not locally convergent at this point.

Since when $x^* = -1$, $|g_1'(x^*)| = \frac{4}{3} > 1$, indicating that Fixed-Point Iteration is not locally convergent at this point.

Since when $x^* = 0$, $|g_1'(x^*)| = \frac{5}{6} < 1$, indicating local convergence at this point.

3. Additional Problem 2 Let $f(x) = -4x^4 + 6x^2 + A$ for some constant A. What value of A should be chosen so that if $x_0 = 1/2$ is the initial approximation, the Newton method produces the sequence $x_1 = -x_0, x_2 = x_0, x_3 = -x_0, \cdots$?

Solution. The Newton Method for finding roots says that:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Given the function $f(x) = -4x^4 + 6x^2 + A$, we can derive that:

$$f'(x) = -16x^3 + 12x$$

Since we want to let the Newton Method produces $x_1 = -x_0$, we apply the Newton Method with $x_0 = \frac{1}{2}$:

$$f(\frac{1}{2}) = \frac{5}{4} + A$$
$$f'(\frac{1}{2}) = 4$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$
$$-x_0 = x_0 - \frac{f(x_0)}{f'(x_0)}$$
$$2x_0 = \frac{f(x_0)}{f'(x_0)}$$
$$2 \cdot \frac{1}{2} = \frac{f(\frac{1}{2})}{f'(\frac{1}{2})}$$
$$1 = \frac{\frac{5}{4} + A}{4}$$
$$A = \frac{11}{4}$$

Check that if $A = \frac{11}{4}$ satisfies $x_2 = x_0$:

$$x_1 = -x_0 = -\frac{1}{2}$$
$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$
$$x_2 = -\frac{1}{2} - \frac{f(-\frac{1}{2})}{f'(-\frac{1}{2})}$$
$$x_2 = -\frac{1}{2} - \frac{4}{-4}$$
$$x_2 = \frac{1}{2} = x_0$$

Therefore, the value of A that should be chosen is $\frac{11}{4}$.

4. Additional Problem 3 Consider the function $f(x) = 54x^6 + 45x^5 - 102x^4 - 69x^3 + 35x^2 + 16x - 4$ on the interval [-2,2].

(a) Apply Newton's method (use newton_err) to find all roots of the function to 6 correct decimal places. Please refer to Notes About Correct Decimal Places in Code page. Report for each root:

- the initial guess and tolerance used;
- the solution with 6 correct decimal places;
- the number of iterations needed;
- the sequence of iterates x_i, the error e_i, and the error ratios $r_q = e_i/e_{i-1}^2$ and $r_l = e_i/e_{i-1}$ (Modify the code to calculate and print these ratios for each iteration.)

(b) Determine the convergence (quadratic or linear) to each of the roots. Compare the limit of the ratios in (a) with the expected value M from Theorem 1.11 or S from Theorem 1.12.

Recall that, for a simple root, Newton's method should converge quadratically. If you observe slower convergence, use the error ratio to decide on the root multiplicity (Theorem 1.12) (you can also find this analytically) and then apply the modified Newton's method. Print out the same information as for the classic Newton's method. NOTE: This means that for a simple root with slower convergence, you should report two sets of information: one for classic Newton's method and one for modified Newton's method.

Solution. (a) By calculator, we can find five true roots.

Root1(X_true): -0.666667

- initial guess = -1, tolerance = 1e-6
- solution = -0.666667
- The number of iterations is 19
- errors = [3.33333000e-01 1.45833000e-01 7.54282411e-02 3.88421203e-02 1.97833741e-02 9.99436679e-03 5.02444632e-03 2.51914282e-03 1.26120591e-03 6.30890346e-04 3.15392494e-04 1.57558137e-04 7.86195485e-05 3.91448951e-05 1.94062280e-05 9.53655851e-06 4.60164015e-06 2.13416323e-06 9.00418211e-07 2.83551003e-07]

  Linear error ratios r_l: [0. 0.51722341 0.51495461 0.50932786 0.5051902 0.50272783 0.5013772 0.50064883 0.50022787 0.4999165 0.4995621 0.49898755 0.49790282

```
C:\Users\13484\AppData\Local\Programs\Python\Python310\python.exe C:\Users\13484\Desktop\newton2.py
Solve the problem f(x)=0 using Newton's method
Enter guess at root: -1
Enter tolerance: 1e-6
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess: x=-1, error=0.333333
Iter 1: x= -0.8125, dx= 0.1875, error = 0.145833, r_l = 0, r_q = 0
Iter 2: x= -0.742095241075, dx= 0.0704047589251, error = 0.0754282410749, r_l = 0.517223406738, r_q = 0
Iter 3: x= -0.705509120304, dx= 0.0365861207712, error = 0.0388421203037, r_l = 0.514954607852, r_q = 6.8270796258
Iter 4: x= -0.686450374126, dx= 0.0190587461775, error = 0.0197833741261, r_l = 0.509327862935, r_q = 13.1127718815
Iter 5: x= -0.676661366785, dx= 0.00978900734098, error = 0.00999436678516, r_l = 0.505190202714, r_q = 25.5360991251
Iter 6: x= -0.671691446315, dx= 0.00496992046977, error = 0.0050244463154, r_l = 0.502727828926, r_q = 50.3011185933
Iter 7: x= -0.669186142818, dx= 0.00250530349722, error = 0.00251914281818, r_l = 0.50137719861, r_q = 99.7875521275
Iter 8: x= -0.667928205914, dx= 0.00125793690446, error = 0.00126120591372, r_l = 0.500648833651, r_q = 198.737773039
Iter 9: x= -0.667297890346, dx= 0.000630315568191, error = 0.000630890345531, r_l = 0.500227868159, r_q = 396.626643371
Iter 10: x= -0.666982392494, dx= 0.000315497851186, error = 0.000315392494345, r_l = 0.499916501464, r_q = 792.398401728
Iter 11: x= -0.666824558137, dx= 0.00015783435723, error = 0.000157558137115, r_l = 0.499562101001, r_q = 1583.93782337
Iter 12: x= -0.666745619549, dx= 7.89385886084e-05, error = 7.86195485065e-05, r_l = 0.498987548001, r_q = 3167.00588835
Iter 13: x= -0.666706144895, dx= 3.94746533701e-05, error = 3.91448951365e-05, r_l = 0.497902822899, r_q = 6333.06642378
Iter 14: x= -0.666686406228, dx= 1.97386671228e-05, error = 1.94062280137e-05, r_l = 0.495753736113, r_q = 12664.5820454
Iter 15: x= -0.666676536559, dx= 9.86966950523e-06, error = 9.53655850844e-06, r_l = 0.491417420311, r_q = 25322.6654848
Iter 16: x= -0.66667160164, dx= 4.93491835497e-06, error = 4.60164015348e-06, r_l = 0.482526285495, r_q = 50597.5279308
Iter 17: x= -0.666669134163, dx= 2.46747692694e-06, error = 2.13416322659e-06, r_l = 0.463783163265, r_q = 100786.490859
Iter 18: x= -0.666667900418, dx= 1.23374501513e-06, error = 9.00418211436e-07, r_l = 0.421906909565, r_q = 197691.959223
The root is -0.666667
The number of iterations is 19
errors = [3.33333000e-01 1.45833000e-01 7.54282411e-02 3.88421203e-02
 1.97833741e-02 9.99436679e-03 5.02444632e-03 2.51914282e-03
 1.26120591e-03 6.30890346e-04 3.15392494e-04 1.57558137e-04
 7.86195485e-05 3.91448951e-05 1.94062280e-05 9.53655851e-06
 4.60164015e-06 2.13416323e-06 9.00418211e-07 2.83551003e-07]
Linear error ratios r_l: [0.         0.51722341 0.51495461 0.50932786 0.5051902  0.50272783
 0.5013772  0.50064883 0.50022787 0.4999165  0.4995621  0.49898755
 0.49790282 0.49575374 0.49141742 0.48252629 0.46378316 0.42190691]
Quadratic error ratios r_q: [0.00000000e+00 6.82707963e+00 1.31127719e+01 2.55360991e+01
 5.03011186e+01 9.97875521e+01 1.98737773e+02 3.96626643e+02
 7.92398402e+02 1.58393782e+03 3.16700589e+03 6.33306642e+03
 1.26645820e+04 2.53226655e+04 5.05975279e+04 1.00786491e+05
 1.97691959e+05]
```

0.49575374 0.49141742 0.48252629 0.46378316 0.42190691]

Quadratic error ratios r_q: [0.00000000e+00 6.82707963e+00 1.31127719e+01 2.55360991e+01 5.03011186e+01 9.97875521e+01 1.98737773e+02 3.96626643e+02 7.92398402e+02 1.58393782e+03 3.16700589e+03 6.33306642e+03 1.26645820e+04 2.53226655e+04 5.05975279e+04 1.00786491e+05 1.97691959e+05]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe C:\Users\13464\Desktop\newton2.py
Solve the problem f(x)=0 using Newton's method
Enter guess at root: 0.8
Enter tolerance: 1e-6
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess: x=0.8, error=0.3
Iter 1: x= 0.560642092747, dx= -0.239357907253, error = 0.0606420927467, r_l = 0, r_q = 0
Iter 2: x= 0.510196211467, dx= -0.05044588128, error = 0.0101962114667, r_l = 0.168137526344, r_q = 0
Iter 3: x= 0.500400032866, dx= -0.00979617860073, error = 0.000400032866017, r_l = 0.0392334807219, r_q = 3.84784886523
Iter 4: x= 0.500000663416, dx= -0.000399369449636, error = 6.63416381053e-07, r_l = 0.00165840468974, r_q = 4.1456710951
The root is 0.500000
The number of iterations is 5
errors = [3.00000000e-01 6.06420927e-02 1.01962115e-02 4.00032866e-04
 6.63416381e-07 1.83042470e-12]
Linear error ratios r_l: [0.         0.16813753 0.03923348 0.0016584 ]
Quadratic error ratios r_q: [0.         3.84784887 4.1456711 ]

Process finished with exit code 0
```

Root2: 0.5

- initial guess = 0.8, tolerance = 1e-6
- solution = 0.500000
- The number of iterations is 5
- errors = [3.00000000e-01 6.06420927e-02 1.01962115e-02 4.00032866e-04 6.63416381e-07 1.83042470e-12]

  Linear error ratios r_l: [0. 0.16813753 0.03923348 0.0016584]

  Quadratic error ratios r_q: [0. 3.84784887 4.1456711]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe C:\Users\13464\Desktop\newton2.py
Solve the problem f(x)=0 using Newton's method
Enter guess at root: -1.5
Enter tolerance: 1e-6
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess: x=-1.5, error=0.118702
Iter 1: x= -1.41859737007, dx= 0.0814026299311, error = 0.0372993700689, r_l = 0, r_q = 0
Iter 2: x= -1.38633173912, dx= 0.0322656309525, error = 0.00503373911639, r_l = 0.134955070477, r_q = 0
Iter 3: x= -1.3814057834, dx= 0.00492595571655, error = 0.000107783399846, r_l = 0.0214121942662, r_q = 4.25373539849
Iter 4: x= -1.38129853212, dx= 0.000107251283316, error = 5.32116529284e-07, r_l = 0.00493690614739, r_q = 45.803956402
The root is -1.381298
The number of iterations is 5
errors = [1.18702000e-01 3.72993701e-02 5.03373912e-03 1.07783400e-04
 5.32116529e-07 4.82044006e-07]
Linear error ratios r_l: [0.         0.13495507 0.02141219 0.00493691]
Quadratic error ratios r_q: [ 0.         4.2537354 45.8039564]

Process finished with exit code 0
```

Root3: -1.381298

- initial guess $= -1.5$, tolerance = 1e-6
- solution $= -1.381298$
- The number of iterations is 5
- errors = [1.18702000e-01 3.72993701e-02 5.03373912e-03 1.07783400e-04 5.32116529e-07 4.82044006e-07]

  Linear error ratios r_l: [0. 0.13495507 0.02141219 0.00493691]

  Quadratic error ratios r_q: [0. 4.2537354 45.8039564]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe C:\Users\13464\Desktop\newton2.py
Solve the problem f(x)=0 using Newton's method
Enter guess at root: 0
Enter tolerance: 1e-6
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess: x=0, error=0.205183
Iter 1: x= 0.25, dx= 0.25, error = 0.044817, r_l = 0, r_q = 0
Iter 2: x= 0.200069832402, dx= -0.0499301675978, error = 0.00511316759777, r_l = 0.11408991226, r_q = 0
Iter 3: x= 0.205146099616, dx= 0.00507626721413, error = 3.69003836334e-05, r_l = 0.00721673657822, r_q = 1.41140231378
Iter 4: x= 0.205182922659, dx= 3.68230427352e-05, error = 7.73408981858e-08, r_l = 0.00209593750987, r_q = 56.7998839983
The root is 0.205183
The number of iterations is 5
errors = [2.05183000e-01 4.48170000e-02 5.11316760e-03 3.69003836e-05
 7.73408982e-08 7.53109524e-08]
Linear error ratios r_l: [0.         0.11408991 0.00721674 0.00209594]
Quadratic error ratios r_q: [ 0.          1.41140231 56.799884  ]

Process finished with exit code 0
```

Root4: 0.205183

- initial guess = 0, tolerance = 1e-6
- solution = 0.205183
- The number of iterations is 5
- errors = [2.05183000e-01 4.48170000e-02 5.11316760e-03 3.69003836e-05 7.73408982e-08 7.53109524e-08]

  Linear error ratios r_l: [0. 0.11408991 0.00721674 0.00209594]

  Quadratic error ratios r_q: [0. 1.41140231 56.799884]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe C:\Users\13464\Desktop\newton2.py
Solve the problem f(x)=0 using Newton's method
Enter guess at root: 2
Enter tolerance: 1e-6
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess: x=2, error=0.823884
Iter 1: x= 1.71291866029, dx= -0.287081339713, error = 0.536802660287, r_l = 0, r_q = 0
Iter 2: x= 1.49139793671, dx= -0.221520723581, error = 0.315281936706, r_l = 0.587333036944, r_q = 0
Iter 3: x= 1.33095570355, dx= -0.160442233152, error = 0.154839703554, r_l = 0.491115048238, r_q = 1.55770119078
Iter 4: x= 1.23014796089, dx= -0.100807742664, error = 0.0540319608897, r_l = 0.348954174217, r_q = 2.25364790947
Iter 5: x= 1.18528473347, dx= -0.0448632274172, error = 0.00916873347247, r_l = 0.169690925917, r_q = 3.14056575261
Iter 6: x= 1.17643597213, dx= -0.00884876134134, error = 0.000319972131135, r_l = 0.0348981821858, r_q = 3.80621623374
Iter 7: x= 1.17611596586, dx= -0.000320006274499, error = 3.41433639139e-08, r_l = 0.000106707305392, r_q = 0.333489373007
The root is 1.176116
The number of iterations is 8
errors = [8.23884000e-01 5.36802660e-01 3.15281937e-01 1.54839704e-01
 5.40319609e-02 9.16873347e-03 3.19972131e-04 3.41433639e-08
 4.42644388e-07]
Linear error ratios r_l: [0.00000000e+00 5.87333037e-01 4.91115048e-01 3.48954174e-01
 1.69690926e-01 3.48981822e-02 1.06707305e-04]
Quadratic error ratios r_q: [0.         1.55770119 2.25364791 3.14056575 3.80621623 0.33348937]

Process finished with exit code 0
```

Root5: 1.176116

- initial guess = 2, tolerance = 1e-6
- solution = 1.176116
- The number of iterations is 8
- errors = [8.23884000e-01 5.36802660e-01 3.15281937e-01 1.54839704e-01 5.40319609e-02 9.16873347e-03 3.19972131e-04 3.41433639e-08 4.42644388e-07]

  Linear error ratios r_l: [0.00000000e+00 5.87333037e-01 4.91115048e-01 3.48954174e-01 1.69690926e-01 3.48981822e-02 1.06707305e-04]

  Quadratic error ratios r_q: [0. 1.55770119 2.25364791 3.14056575 3.80621623 0.33348937]

Given the detailed solution and referencing Theorem 1.11 and Theorem 1.12, we analyze the convergence behavior for each root and compare the observed error ratios to the expected theoretical values.

(b)
- **Theorem 1.11** states that if f is twice continuously differentiable and $f(r) = 0$ with $f'(r) \neq 0$, then Newton's Method is locally and quadratically convergent to r. The quadratic convergence rate (M) is defined by $\lim_{i\to\infty} \frac{e_{i+1}}{e_i^2} = M$, where $M = \frac{f''(r)}{2f'(r)}$.

- **Theorem 1.12** addresses the case where the root r of f has multiplicity $m > 1$. It states that Newton's Method is locally convergent to r, with the linear convergence rate (S) given by $\lim_{i\to\infty} \frac{e_{i+1}}{e_i} = S$, where $S = \frac{m-1}{m}$.

Analysis and Calculations. For the function $f(x) = 54x^6 + 45x^5 - 102x^4 - 69x^3 + 35x^2 + 16x - 4$, the derivatives are:

$$f'(x) = 324x^5 + 225x^4 - 408x^3 - 207x^2 + 70x + 16,$$
$$f''(x) = 1620x^4 + 900x^3 - 1224x^2 - 414x + 70.$$

Calculation for Root 1: -0.666667. Substituting $x = -0.666667$ into the derivatives to calculate M:

$$M = \frac{f''(-0.666667)}{2f'(-0.666667)}$$
$$M \to \infty$$
$$S = \frac{m-1}{m} = \frac{2-1}{2} = \frac{1}{2}$$

Since we observe observe slower convergence, $S = \frac{1}{2}$,we need to use modified newton function to solve this. Specifically, the modified code generate:

Iter 1: x= -0.625, dx= 0.375, error = 0.041667, r_l = 0, r_q = 0

Iter 2: x= -0.66785079929, dx= -0.0428507992895, error = 0.00118379928952, r_l = 0.0284109556608, r_q = 0

Iter 3: x= -0.666667467024, dx= 0.00118333226588, error = 4.67023645312e-07, r_l = 0.000394512523741, r_q = 0.333259638888

The root is -0.666667

The number of iterations is 4

errors = [3.33333000e-01 4.16670000e-02 1.18379929e-03 4.67023645e-07 3.33390747e-07]

Linear error ratios r_l: [0. 0.02841096 0.00039451]

Quadratic error ratios r_q: [0. 0.33325964]

Since M $\to \infty$ and $S = 0$, it is linear convergence.

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe C:\Users\13464\Desktop\newton2.py
Solve the problem f(x)=0 using Newton's method
Enter guess at root: -1
Enter tolerance: 1e-6
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess: x=-1, error=0.333333
Iter 1: x= -0.625, dx= 0.375, error = 0.041667, r_l = 0, r_q = 0
Iter 2: x= -0.66785079929, dx= -0.0428507992895, error = 0.00118379928952, r_l = 0.0284109556608, r_q = 0
Iter 3: x= -0.666667467024, dx= 0.00118333226588, error = 4.67023645312e-07, r_l = 0.000394512523741, r_q = 0.333259638888
The root is -0.666667
The number of iterations is 4
errors = [3.33333000e-01 4.16670000e-02 1.18379929e-03 4.67023645e-07
 3.33390747e-07]
Linear error ratios r_l: [0.         0.02841096 0.00039451]
Quadratic error ratios r_q: [0.         0.33325964]

Process finished with exit code 0
```

Calculation for Root 2: 0.5. Substituting $x = 0.5$ into the derivatives to calculate M:

$$M = \frac{f''(0.5)}{2f'(0.5)}$$

$$M = \frac{-229.25}{-27.5625 \cdot 2} \to 4.158730$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since $M < \infty$ and $S = 0$, this is quadratic convergence. We observe that the generated r_q = 4.1456711, which is close to the M we calculated 4.158730.

Calculation for Root 3: -1.381298. Substituting $x = -0.666667$ into the derivatives to calculate M:

$$M = \frac{f''(-1.381298)}{2f'(-1.381298)}$$

$$M = \frac{1831.99330298}{-210.499160747 \cdot 2} \to -4.351545$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since $M < \infty$ and $S = 0$, this is quadratic convergence. We observe that the generated r_q = 4.2537354 , which is close to the M we calculated 4.158730.

Calculation for Root 4: 0.205183. Substituting $x = 0.205183$ into the derivatives to calculate M:

$$M = \frac{f''(0.205183)}{2f'(0.205183)}$$

$$M = \frac{-55.8305312562}{18.6403265585 \cdot 2} \to -1.457974$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since $M < \infty$ and $S = 0$, this is quadratic convergence. We observe that the generated r_q = 1.41140231, which is close to the M we calculated -1.457974.

Calculation for Root 5: $r = 1.176116$**.** Substituting $x = 1.176116$ into the derivatives to calculate M:

$$M = \frac{f''(1.176116)}{2f'(1.176116)}$$

$$M = \frac{2453.83771527}{307.860799265 \cdot 2} \rightarrow 3.985304$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since $M < \infty$ and $S = 0$, this is quadratic convergence. We observe that the generated r_q = 3.80621623, which is close to the M we calculated -1.457974.

5. **Additional Problem 4** Consider the function in AP3.

(a) Apply Newton-Bisection Method (use newtonBisection_err) to find all roots of the function to 6 correct decimal places. Please refer to Notes About Correct Decimal Places in Code page.

- the initial guess and tolerance used;
- the solution with 6 correct decimal places;
- the number of iterations needed;
- the sequence of iterates x_i, the error e_i, and the error ratios $r_q = e_i/e_{i-1}^2$ and $r_l = e_i/e_{i-1}$ (Modify the code to calculate and print these ratios for each iteration.)

(b) Determine the convergence (quadratic, superlinear, or linear) to each of the roots.

Solution. (a) **Root1(X_true):** -0.666667

- initial guess a = -1.5, b = 0.2, tolerance = 5e-7
- solution = -0.666667
- The number of iterations is 15
- errors = [1.66666667e-02 8.24780574e-03 4.10368670e-03 2.04692639e-03 1.02225059e-03 5.10824194e-04 2.55337077e-04 1.27649815e-04 6.38202307e-05 3.19089466e-05 1.59541821e-05 7.97701649e-06 3.98849237e-06 1.99424606e-06 9.97115497e-07 4.98538067e-07]

  Linear error ratios r_l: [0.49486834 0.4975489 0.49880182 0.4994076 0.49970545 0.49985314 0.49992667 0.49996336 0.49998169 0.49999087 0.49999533 0.49999801 0.49999997 0.49999622 0.49998026]

  Quadratic error ratios r_q: [2.96921007e+01 6.03250026e+01 1.21549683e+02 2.43979267e+02 4.88828726e+02 9.78522833e+02 1.95790865e+03 3.91667911e+03 7.83421935e+03 1.56693006e+04 3.13394522e+04 6.26798265e+04 1.25360643e+05 2.50719423e+05 5.01426628e+05]

  Superlinear error ratios r_sl [0.00000000e+00 2.98529342e+01 6.04769115e+01 1.21697302e+02 2.44124780e+02 4.88973196e+02 9.78666784e+02 1.95805235e+03 3.91682265e+03 7.83436332e+03 1.56694401e+04 3.13396204e+04 6.26800722e+04 1.25359704e+05 2.50711420e+05]

```
Monitor iterations? (1/0): 1
Interval = [-1.500000,0.200000], guess x = -0.650000, error = 0.016667
Iter 1: x= -0.658418860924, dx= -0.00841886092367, error = 0.00824780574299, interval = [-1.5,-0.658418860924], Newton? 1
Iter 2: x= -0.662562979968, dx= -0.00414411904445, error = 0.00410368669855, interval = [-1.5,-0.662562979968], Newton? 1
Iter 3: x= -0.664619740282, dx= -0.00205676031347, error = 0.00204692638508, interval = [-1.5,-0.664619740282], Newton? 1
Iter 4: x= -0.665644416077, dx= -0.00102467579499, error = 0.00102225059009, interval = [-1.5,-0.665644416077], Newton? 1
Iter 5: x= -0.666155842472, dx= -0.000511426395616, error = 0.000510824194478, interval = [-1.5,-0.666155842472], Newton? 1
Iter 6: x= -0.66641132959, dx= -0.000255487117927, error = 0.000255337076551, interval = [-1.5,-0.66641132959], Newton? 1
Iter 7: x= -0.666539016852, dx= -0.000127687261714, error = 0.000127649814837, interval = [-1.5,-0.666539016852], Newton? 1
Iter 8: x= -0.666602846436, dx= -6.38295841522e-05, error = 6.38202306846e-05, interval = [-1.5,-0.666602846436], Newton? 1
Iter 9: x= -0.66663475772, dx= -3.19112841313e-05, error = 3.19089465534e-05, interval = [-1.5,-0.66663475772], Newton? 1
Iter 10: x= -0.666650712485, dx= -1.59547644749e-05, error = 1.59541820784e-05, interval = [-1.5,-0.666650712485], Newton? 1
Iter 11: x= -0.66665868965, dx= -7.97716558743e-06, error = 7.97701649102e-06, interval = [-1.5,-0.66665868965], Newton? 1
Iter 12: x= -0.666662678174, dx= -3.98852412398e-06, error = 3.98849236705e-06, interval = [-1.5,-0.666662678174], Newton? 1
Iter 13: x= -0.666664672421, dx= -1.99424630465e-06, error = 1.9942460624e-06, interval = [-1.5,-0.666664672421], Newton? 1
Iter 14: x= -0.666665669551, dx= -9.97130565339e-07, error = 9.97115497059e-07, interval = [-1.5,-0.666665669551], Newton? 1
Iter 15: x= -0.666666168129, dx= -4.98577429897e-07, error = 4.98538067162e-07, interval = [-1.5,-0.666666168129], Newton? 1
The root is -0.666666.
The number of iterations is 15
errors = [1.66666667e-02 8.24780574e-03 4.10368670e-03 2.04692639e-03
 1.02225059e-03 5.10824194e-04 2.55337077e-04 1.27649815e-04
 6.38202307e-05 3.19089466e-05 1.59541821e-05 7.97701649e-06
 3.98849237e-06 1.99424606e-06 9.97115497e-07 4.98538067e-07]
Linear error ratios r_l: [0.49486834 0.4975489  0.49880182 0.4994076  0.49970545 0.49985314
 0.49992667 0.49996336 0.49998169 0.49999087 0.49999533 0.49999801
 0.49999997 0.49999622 0.49998026]
Quadratic error ratios r_q: [2.96921007e+01 6.03250026e+01 1.21549683e+02 2.43979267e+02
 4.88828726e+02 9.78522833e+02 1.95790865e+03 3.91667911e+03
 7.83421935e+03 1.56693006e+04 3.13394522e+04 6.26798265e+04
 1.25360643e+05 2.50719423e+05 5.01426628e+05]
Superlinear error ratios r_sl [0.00000000e+00 2.98529342e+01 6.04769115e+01 1.21697302e+02
 2.44124780e+02 4.88973196e+02 9.78666784e+02 1.95805235e+03
 3.91682265e+03 7.83436332e+03 1.56694401e+04 3.13396204e+04
 6.26800722e+04 1.25359704e+05 2.50711420e+05]
```

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\newtonBisection (1).py"
Solve the problem f(x)=0 on interval [a,b] using Newton-Bisection method
Enter a: 0.3
Enter b: 1
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Interval = [0.300000,1.000000], guess x = 0.650000, error = 0.150000
Iter 1: x= 0.539200874286, dx= -0.110799125714, error = 0.0392008742858, interval = [0.3,0.539200874286], Newton? 1
Iter 2: x= 0.504851571098, dx= -0.0343493031873, error = 0.00485157109849, interval = [0.3,0.504851571098], Newton? 1
Iter 3: x= 0.500094275397, dx= -0.00475729570174, error = 9.42753967551e-05, interval = [0.3,0.500094275397], Newton? 1
Iter 4: x= 0.500000036935, dx= -9.42384619975e-05, error = 3.69347575857e-08, interval = [0.3,0.500000036935], Newton? 1
Iter 5: x= 0.5, dx= -3.69347518125e-08, error = 5.77315972805e-15, interval = [0.3,0.5], Newton? 1
The root is 0.500000.
The number of iterations is 5
errors = [1.50000000e-01 3.92008743e-02 4.85157110e-03 9.42753968e-05
 3.69347576e-08 5.77315973e-15]
Linear error ratios r_l: [2.61339162e-01 1.23761809e-01 1.94319314e-02 3.91775149e-04
 1.56306961e-07]
Quadratic error ratios r_q: [1.74226108 3.15711858 4.00528633 4.15564572 4.23197475]
Superlinear error ratios r_sl: [0.         0.82507872 0.49570148 0.08075222 0.00165798]

Process finished with exit code 0
```

Root2: 0.5

- initial guess a = 0.3, b = 1, tolerance = 5e-7
- solution = 0.500000
- The number of iterations is 5
- errors = [1.50000000e-01 3.92008743e-02 4.85157110e-03 9.42753968e-05 3.69347576e-08 5.77315973e-15]

  Linear error ratios r_l: [2.61339162e-01 1.23761809e-01 1.94319314e-02 3.91775149e-04 1.56306961e-07]

  Quadratic error ratios r_q: [1.74226108 3.15711858 4.00528633 4.15564572 4.23197475]

  Superlinear error ratios r_sl: [0. 0.82507872 0.49570148 0.08075222 0.00165798]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\newtonBisection (1).py"
Solve the problem f(x)=0 on interval [a,b] using Newton-Bisection method
Enter a: -2
Enter b: -1
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Interval = [-2.000000,-1.000000], guess x = -1.500000, error = 0.118702
Iter 1: x= -1.41859737007, dx= 0.0814026299311, error = 0.0372993700689, interval = [-1.41859737007,-1], Newton? 1
Iter 2: x= -1.38633173912, dx= 0.0322656309525, error = 0.00503373911639, interval = [-1.38633173912,-1], Newton? 1
Iter 3: x= -1.3814057834, dx= 0.00492595571655, error = 0.000107783399846, interval = [-1.3814057834,-1], Newton? 1
Iter 4: x= -1.38129853212, dx= 0.000107251283316, error = 5.32116529284e-07, interval = [-1.38129853212,-1], Newton? 1
Iter 5: x= -1.38129848204, dx= 5.00725236829e-08, error = 4.82044005601e-07, interval = [-1.38129848204,-1], Newton? 1
The root is -1.381298.
The number of iterations is 5
errors = [1.18702000e-01 3.72993701e-02 5.03373912e-03 1.07783400e-04
 5.32116529e-07 4.82044006e-07]
Linear error ratios r_l: [0.31422697 0.13495507 0.02141219 0.00493691 0.90589933]
Quadratic error ratios r_q: [2.64719190e+00 3.61815951e+00 4.25373540e+00 4.58039564e+01
 1.70244538e+06]
Superlinear error ratios r_sl: [0.00000000e+00 1.13692331e+00 5.74063160e-01 9.80763213e-01
 8.40481306e+03]

Process finished with exit code 0
```

Root3: -1.381298

- initial guess a $= -2$, b $= -1$, tolerance = 5e-7
- solution $= -1.381298$
- The number of iterations is 5
- errors = [1.18702000e-01 3.72993701e-02 5.03373912e-03 1.07783400e-04 5.32116529e-07 4.82044006e-07]

  Linear error ratios r_l: [0.31422697 0.13495507 0.02141219 0.00493691 0.90589933]

  Quadratic error ratios r_q: [2.64719190e+00 3.61815951e+00 4.25373540e+00 4.58039564e+01 1.70244538e+06]

  Superlinear error ratios r_sl: [0.00000000e+00 1.13692331e+00 5.74063160e-01 9.80763213e-01 8.40481306e+03]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\newtonBisection (1).py"
Solve the problem f(x)=0 on interval [a,b] using Newton-Bisection method
Enter a: 0
Enter b: 0.5
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Interval = [0.000000,0.500000], guess x = 0.250000, error = 0.044817
Iter 1: x= 0.200069832402, dx= -0.0499301675978, error = 0.00511316759777, interval = [0.200069832402,0.25], Newton? 1
Iter 2: x= 0.205146099616, dx= 0.00507626721413, error = 3.69003836334e-05, interval = [0.205146099616,0.25], Newton? 1
Iter 3: x= 0.205182922659, dx= 3.68230427352e-05, error = 7.73408981858e-08, interval = [0.205182922659,0.25], Newton? 1
Iter 4: x= 0.205182924689, dx= 2.02994576703e-09, error = 7.53109524188e-08, interval = [0.205182924689,0.25], Newton? 1
The root is 0.205183.
The number of iterations is 4
errors = [4.48170000e-02 5.11316760e-03 3.69003836e-05 7.73408982e-08
 7.53109524e-08]
Linear error ratios r_l: [0.11408991 0.00721674 0.00209594 0.97375327]
Quadratic error ratios r_q: [2.54568383e+00 1.41140231e+00 5.67998840e+01 1.25904055e+07]
Superlinear error ratios r_sl: [0.00000000e+00 1.61026766e-01 4.09909800e-01 2.63887031e+04]

Process finished with exit code 0
```

Root4: 0.205183

- initial guess a = 0, b = 0.5, tolerance = 5e-7
- solution = 0.205183
- The number of iterations is 4
- errors = [4.48170000e-02 5.11316760e-03 3.69003836e-05 7.73408982e-08 7.53109524e-08]

  Linear error ratios r_l: [0.11408991 0.00721674 0.00209594 0.97375327]

  Quadratic error ratios r_q: [2.54568383e+00 1.41140231e+00 5.67998840e+01 1.25904055e+07]

  Superlinear error ratios r_sl: [0.00000000e+00 1.61026766e-01 4.09909800e-01 2.63887031e+04]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\newtonBisection (1).py"
Solve the problem f(x)=0 on interval [a,b] using Newton-Bisection method
Enter a: 1
Enter b: 2
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Interval = [1.000000,2.000000], guess x = 1.500000, error = 0.323884
Iter 1: x= 1.33686715707, dx= -0.163132842925, error = 0.160751607075, interval = [1,1.33686715707], Newton? 1
Iter 2: x= 1.23340618877, dx= -0.103460968308, error = 0.0572906387673, interval = [1,1.23340618877], Newton? 1
Iter 3: x= 1.18629406042, dx= -0.0471121283496, error = 0.0101785104176, interval = [1,1.18629406042], Newton? 1
Iter 4: x= 1.17650850608, dx= -0.00978555433473, error = 0.000392956082876, interval = [1,1.17650850608], Newton? 1
Iter 5: x= 1.17611617152, dx= -0.00039233456477, error = 6.21518105648e-07, interval = [1,1.17611617152], Newton? 1
Iter 6: x= 1.17611555736, dx= -6.1416165531e-07, error = 7.35645033778e-09, interval = [1,1.17611555736], Newton? 1
Iter 7: x= 1.17611555735, dx= -1.50301993074e-12, error = 7.35494731785e-09, interval = [1,1.17611555735], Newton? 1
The root is 1.176116.
The number of iterations is 7
errors = [3.23884450e-01 1.60751607e-01 5.72906388e-02 1.01785104e-02
 3.92956083e-04 6.21518106e-07 7.35645034e-09 7.35494732e-09]
Linear error ratios r_l: [0.49632394 0.35639232 0.17766446 0.03860644 0.00158165 0.01183626
 0.99979569]
Quadratic error ratios r_q: [1.53241053e+00 2.21703739e+00 3.10110804e+00 3.79293644e+00
 4.02499878e+00 1.90441125e+04 1.35907352e+08]
Superlinear error ratios r_sl: [0.00000000e+00 1.10036874e+00 1.10521110e+00 6.73870005e-01
 1.55390886e-01 3.01210777e+01 1.60863485e+06]

Process finished with exit code 0
```

Root5: 1.176116

- initial guess a = 1, b = 2, tolerance = 5e-7
- solution = 1.176116
- The number of iterations is 7
- errors = [3.23884450e-01 1.60751607e-01 5.72906388e-02 1.01785104e-02 3.92956083e-04 6.21518106e-07 7.35645034e-09 7.35494732e-09]

  Linear error ratios r_l: [0.49632394 0.35639232 0.17766446 0.03860644 0.00158165 0.01183626 0.99979569]

  Quadratic error ratios r_q: [1.53241053e+00 2.21703739e+00 3.10110804e+00 3.79293644e+00 4.02499878e+00 1.90441125e+04 1.35907352e+08]

  Superlinear error ratios r_sl: [0.00000000e+00 1.10036874e+00 1.10521110e+00 6.73870005e-01 1.55390886e-01 3.01210777e+01 1.60863485e+06]

(b) **Calculation for Root 1:** -0.666667 Substituting $x = -0.666667$ into the derivatives to calculate M:

$$M = \frac{f''(-0.666667)}{2f'(-0.666667)}$$
$$M \to \infty$$

$$S = \frac{m-1}{m} = \frac{2-1}{2} = \frac{1}{2}$$

Since r_q = 5.01426628e+05 and r_sl = 2.50711420e+05 diverge to ∞ and r_l = 0.49998026 $\approx$ 0.5, this is linear convergence.

Calculation for Root 2: 0.5**.** Substituting $x = 0.5$ into the derivatives to calculate M:

$$M = \frac{f''(0.5)}{2f'(0.5)}$$

$$M = \frac{-229.25}{-27.5625 \cdot 2} \rightarrow 4.158730$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since r_sl = 0.00165798 and r_l = 1.56306961e-07 converge 0, r_q = 4.23197475 $\approx$ 4.158730, this is quadratic convergence.

Calculation for Root 3: -1.381298**.** Substituting $x = -0.666667$ into the derivatives to calculate M:

$$M = \frac{f''(-1.381298)}{2f'(-1.381298)}$$

$$M = \frac{1831.99330298}{-210.499160747 \cdot 2} \rightarrow -4.351545$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since r_sl = 9.80763213e-01 and r_l = 0.00493691 converge to 0, r_q = 4.25373540e+00$\approx$ -4.351545, this is quadratic convergence. (except the last several value in the array)

Calculation for Root 4: 0.205183**.** Substituting $x = 0.205183$ into the derivatives to calculate M:

$$M = \frac{f''(0.205183)}{2f'(0.205183)}$$

$$M = \frac{-55.8305312562}{18.6403265585 \cdot 2} \rightarrow -1.457974$$

$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since r_sl = 4.09909800e-01 and r_l = 0.00209594 converge to 0, r_q = 1.41140231e+00$\approx$ -4.351545, this is quadratic convergence. (except the last several value in the array)

Calculation for Root 5: $r = 1.176116$. Substituting $x = 1.176116$ into the derivatives to calculate M:

$$M = \frac{f''(1.176116)}{2f'(1.176116)}$$
$$M = \frac{2453.83771527}{307.860799265 \cdot 2} \to 3.985304$$
$$S = \frac{m-1}{m} = \frac{1-1}{1} = 0$$

Since r_sl = 1.55390886e-01 and r_l = 0.01183626 converge to 0, r_q = 4.02499878e+00≈ 3.985304, this is quadratic convergence. (except the last several value in the array)

1. Newton Method

```
   #!/usr/bin/env python3
'''
 NEWTON'S METHOD

 Solves the problem f(x)=0 using Newton's method. For a known tru solution
 calculates errors.

 The main function is newton:

  [state,x,errors,iters] = newton(x0, tolerance, maxIteration, debug)

  Inputs:
    x0              The initial guess at the solution
    tolerance       The convergence tolerance (must be > 0).
    maxIteration    The maximum number of iterations that can be taken.
    debug           Boolean to set debugging output
  Outputs:
    x               The solution
    errors          Array with errors at each iteration
    iter            number of iterations to convergence
  Return:
    state           An error status code.
      SUCCESS       Sucessful termination.
```

```
      WONT_STOP     Error: Exceeded maximum number of iterations.
      BAD_ITERATE   Error: The function had a vanishing derivative

  Remark: We assume that we known the two functions
    f               The name of the function for which a root is sought
    df              The name of th derivative of the function. The derivative
                    of f must be computed by hand and coded correctly as df, or
                    newton will not work!
'''
from numpy import zeros, abs
############################### VARIABLES ##############################
SUCCESS = 0
WONT_STOP = 1
BAD_ITERATE = 2
x_true = -0.666667
############################### FUNCTIONS ##############################
# The function for which a root is sought
def f(x):
    return 54*x**6 + 45*x**5 - 102*x**4 - 69*x**3 + 35*x**2 + 16*x - 4

# The name of th derivative of the function. The derivative of f must
# be computed by hand and coded correctly as df, or newton will not work!
def df(x):
    return 324*x**5 + 225*x**4 - 408*x**3 - 207*x**2 + 70*x + 16

def newton(x0,TOL,MAX_ITERS,debug):
    global x_true, SUCCESS, WONT_STOP, BAD_ITERATE
    prec = 12
    eps = 1e-20
    # formatting string, this decides how output will look
    fmt = f"Iter %d: x= %.{prec}g, dx= %.{prec}g, error = %.{prec}g, r_l = %.{prec}g, r_

    errors = zeros(MAX_ITERS+1)
```

```
ratios_l = zeros(MAX_ITERS)
ratios_q = zeros(MAX_ITERS)
x = x0
err = abs(x - x_true)
errors[0] = err
prev_error = err

if debug:
    print("Guess: x=%.8g, error=%.8g"%(x,err))

## Newton Loop
for itn in range(1,MAX_ITERS+1):
    dfx = df(x)
    if(abs(dfx) < eps):
        state = BAD_ITERATE
        iters = itn
        return state, x, errors[:itn], iters, ratios_l[:itn-1], ratios_q[:itn-2]

    dx = -f(x)/dfx
    # Use for Modified Netwon, multiplicity 2
    dx = 2*dx
    x += dx
    err = abs(x - x_true)
    errors[itn] = err

    # Check error tolerance
    if (abs(dx) <= TOL):
        iter = itn
        state = SUCCESS
        return state, x, errors[:itn+1], iter, ratios_l[:itn-1], ratios_q[:itn-2]

    if itn > 1:  # Ratios start from the second iteration
        ratios_l[itn - 1] = err / prev_error
```

```python
        if itn > 2:  # Quadratic ratios require at least two previous errors
            ratios_q[itn - 2] = err / (prev_error ** 2)

        prev_error = err

        if debug:
            print(fmt % (itn, x, dx, err, ratios_l[itn - 1] if itn > 1 else 0, ratios_q[

    state = WONT_STOP
    iter = itn
    return state, x, errors[:itn], iter, ratios_l[:itn-1], ratios_q[:itn-2]


################################ MAIN ###############################

###input
print("Solve the problem f(x)=0 using Newton's method")
x0 = float(input("Enter guess at root: "))
tol = float(input("Enter tolerance: "))
maxIter = int(input("Enter maxIteration: "))
debug = bool(input("Monitor iterations? (1/0): "))

### Solve
[s,x,errors,iters, ratios_l, ratios_q] = newton(x0,tol,maxIter,debug)
if s == SUCCESS:
    print(f"The root is {x:.6f}")
    print("The number of iterations is %d"%(iters))
    errors = errors[:iters+1]
    print("errors =",errors)
    print("Linear error ratios r_l:", ratios_l)
    print("Quadratic error ratios r_q:", ratios_q)
    exit()
elif s == WONT_STOP:
```

```
    print("ERROR: Failed to converge in %d iterations!"%(maxIter))
elif s == BAD_ITERATE:
    print("ERROR: Obtained a vanishing derivative!")
else:
    print("ERROR: Coding error!")
exit(1) #technically not necessary, but good for general practice
```

2.Newton Bisection Method

```
#!/usr/bin/env python3
'''
 NEWTON-BISECTION METHOD

 Solves the problem f(x)=0 using Newton-Bisection method. For a known true solution
 calculates errors.

 The main function is newton:

  [state,x,errors,iters] = newtonBIsection(x0, tolerance, maxIteration, debug)

  Inputs:
    a,b              The initial bounding interval, with a root between.
    tolerance        The convergence tolerance (must be > 0).
    maxIteration     The maximum number of iterations that can be taken.
    debug            Boolean to set debugging output
  Outputs:
    x                The solution
    errors           Array with errors at each iteration
    iter             number of iterations to convergence
  Return:
    state            An error status code.
      SUCCESS        Sucessful termination.
      WONT_STOP      Error: Exceeded maximum number of iterations.
      BAD_DATA       Error: The interval may not bracket a root
```

```
  Remark: We assume that we known the two functions
    f               The name of the function for which a root is sought
    df              The name of the derivative of the function.
'''
import math
from numpy import zeros,sign

############################## VARIABLES ##############################
SUCCESS = 0
WONT_STOP = 1
BAD_DATA = 2
x_true = 1.17611555
############################## FUNCTIONS ##############################
# The function for which a root is sought
def f(x):
    return 54*x**6 + 45*x**5 - 102*x**4 - 69*x**3 + 35*x**2 + 16*x - 4

# The name of the derivative of the function.
def df(x):
    return 324*x**5 + 225*x**4 - 408*x**3 - 207*x**2 + 70*x + 16

def newtonBisection(a,b,TOL,MAX_ITERS,debug):
    global x_true, SUCCESS, WONT_STOP, BAD_DATA
    prec = 12
    eps = 1e-20
    # formatting string, this decides how output will look
    fmt = f"Iter %d: x= %.{prec}g, dx= %.{prec}g, error = %.{prec}g, "
    fmt += f"interval = [%.{prec}g,%.{prec}g], Newton? %d"

    # Swap a and b if necessary so a < b
    if (a>b):
        c = a
```

```
    a = b
    b = c
fa = f(a)
fb = f(b)

# Make sure there is a root between a and b
if(sign(fa)*sign(fb) > 0.0):
    state = BAD_DATA
    x = None
    errors = None
    iter = 0
    return state,x,errors,iter,[],[],[]

errors = zeros(MAX_ITERS+1)
x = a+(b-a)/2
err = abs(x - x_true)
errors[0] = err
ratios_l = zeros(MAX_ITERS)
ratios_q = zeros(MAX_ITERS)
ratios_sl = zeros(MAX_ITERS)

if debug:
    print("Interval = [%f,%f], guess x = %f, error = %f"%(a,b,x,err))

fx = f(x)
if(sign(fa)*sign(fx) > 0.0):
    a = x
else:
    b = x

## NewtonBisection Loop
for itn in range(1,MAX_ITERS+1):
    dfx = df(x)
```

```
usedNewton = True
if(abs(dfx) > eps):
    xNew = x - fx/dfx # Newton
    if(xNew < a or b < xNew):
        xNew = a + (b-a)/2 # Revert to Bisection
        usedNewton = False
else:
    xNew = a + (b-a)/2;  # Revert to Bisection
    usedNewton = False

fx = f(xNew)
if(sign(fa)*sign(fx) > 0.0):
    a = xNew
else:
    b = xNew

dx = xNew - x
x = xNew
err = abs(x - x_true)
errors[itn] = err
ratios_l[itn - 1] = err / errors[itn - 1]
ratios_q[itn - 1] = err / (errors[itn - 1] ** 2)
if itn > 1:
    ratios_sl[itn - 1] = err/ (errors[itn - 1] * errors[itn - 2])

if debug:
    print(fmt % (itn, x, dx,err,a,b,usedNewton))

# Check error tolerance
if (abs(dx) <= TOL):
    iter = itn
    state = SUCCESS
    return state, x, errors, iter, ratios_l[:itn], ratios_q[:itn], ratios_sl[:it
```

```
    state = WONT_STOP
    iter = itn
    return state, x, errors, MAX_ITERS, ratios_l[:MAX_ITERS], ratios_q[:MAX_ITERS], rati


################################ MAIN ###############################

###input
print("Solve the problem f(x)=0 on interval [a,b] using Newton-Bisection method")
a = float(input("Enter a: "))
b = float(input("Enter b: "))
tol = float(input("Enter tolerance: "))
maxIter = int(input("Enter maxIteration: "))
debug = bool(input("Monitor iterations? (1/0): "))

### Solve
s, x, errors, iters, r_l, r_q, r_sl = newtonBisection(a, b, tol, maxIter, debug)
if s == SUCCESS:
    print(f"The root is {x:.6f}.")
    print("The number of iterations is %d"%(iters))
    errors = errors[:iters+1]
    print("errors =",errors)
    if debug:  # Optionally print error ratios if debugging is enabled
        print("Linear error ratios r_l:", r_l)
        print("Quadratic error ratios r_q:", r_q)
        print("Superlinear error ratios r_sl:", r_sl)
    exit()
elif s == WONT_STOP:
    print("ERROR: Failed to converge in %d iterations!"%(maxIter))
elif s == BAD_DATA:
    print("ERROR: Unsuitable interval!")
else:
```

```
    print("ERROR: Coding error!")
exit(1) #technically, not necessary but good for general practice
```