M 348 HOMEWORK 4

ZUN CAO

1. Chapter 1.4 Ex # 2 Apply two steps of Newton's Method with initial guess $x_0 = 1$.

(a) $x^3 + x^2 - 1 = 0$

(b) $x^2 + \frac{1}{x+1} - 3x = 0$

(c) $5x - 10 = 0$

Solution. See Page2.

2. Additional Problem 1 Apply two steps of the Secant Method for the functions from Sec 1.4 Ex # 2 with initial guesses $x_0 = 0.5$ and $x_1 = 1.0$.

Solution. See Page3.

Date: 2/16/2024.

M348HW4

1.4 #2 We know that the Newton Method is

x_0 = initial guess

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \text{for } i = 0,1,2,\cdots$$

(a) Since $x_0 = 1$, we have

Step 1: $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ $\qquad f(x_0) = x_0^3 + x_0^2 - 1 = 1^3 + 1^2 - 1 = 1$

$f'(x_0) = 3x_0^2 + 2x_0 = 3\cdot 1^2 + 2\cdot 1 = 5$

$x_1 = 1 - \frac{1}{5} = \frac{4}{5}$

Step 2: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ $\qquad f(x_1) = x_1^3 + x_1^2 - 1 = (\frac{4}{5})^3 + (\frac{4}{5})^2 - 1 = \frac{19}{125}$

$f'(x_1) = 3x_1^2 + 2x_1 = 3(\frac{4}{5})^2 + 2(\frac{4}{5}) = \frac{88}{25}$

$x_2 = \frac{4}{5} - \frac{\frac{19}{125}}{\frac{88}{25}} = \frac{333}{440} \approx \boxed{0.756818}$

(b) Since $x_0 = 1$, we have

Step 1: $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ $\qquad f(x_0) = x_0^2 + \frac{1}{x_0+1} - 3x_0 = 1^2 + \frac{1}{1+1} - 3 = -\frac{3}{2}$

$f'(x_0) = 2x_0 - (x_0+1)^{-2} - 3 = 2\cdot 1 - 2^{-2} - 3 = -\frac{5}{4}$

$x_1 = 1 - \frac{(-\frac{3}{2})}{(-\frac{5}{4})} = -\frac{1}{5}$

Step 2: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ $\qquad f(x_1) = x_1^2 + \frac{1}{x_1+1} - 3x_1 = (-\frac{1}{5})^2 + \frac{1}{1-\frac{1}{5}} - 3(-\frac{1}{5}) = \frac{189}{100}$

$f'(x_1) = 2x_1 - (x_1+1)^{-2} - 3 = 2(-\frac{1}{5}) - (1-\frac{1}{5})^{-2} - 3 = -\frac{397}{80}$

$x_2 = -\frac{1}{5} - \frac{\frac{189}{100}}{-\frac{397}{80}} = \frac{359}{1985} \approx \boxed{0.180856}$

(c) Since $x_0 = 1$, we have

Step 1: $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ $\qquad f(x_0) = 5x_0 - 10 = 5\cdot 1 - 10 = -5$

$f'(x_0) = 5$

$x_1 = 1 - \frac{-5}{5} = 2$

Step 2: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ $\qquad f(x_1) = 5x_1 - 10 = 5\cdot 2 - 10 = 0$

$f'(x_1) = 5$

$x_2 = 2 - 0 = \boxed{2}$

AD1. We know the secant method is:

x_0, x_1 = initial guess

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \quad \text{for } i = 1, 2, 3 \cdots$$

Since $x_0 = 0.5$, $x_1 = 1.0$, we have:

(a) $$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

$f(x_1) = x_1^3 + x_1^2 - 1 = 1^3 + 1^2 - 1 = 1$

$f(x_0) = x_0^3 + x_0^2 - 1 = (0.5)^3 + (0.5)^2 - 1 = -0.625$

Step 1: $$x_2 = 1.0 - \frac{1(1-0.5)}{1-(-0.625)} = \frac{9}{13} \approx 0.692308$$

Step 2: $$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$f(x_1) = 1$

$f(x_2) = x_2^3 + x_2^2 - 1 = (\frac{9}{13})^3 + (\frac{9}{13})^2 - 1 = -\frac{415}{2197}$

$$x_3 = \frac{9}{13} - \frac{(-\frac{415}{2197})(\frac{9}{13} - 1)}{(-\frac{415}{2197}) - 1} = \frac{484}{653} \approx \boxed{0.741194}$$

(b) $$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

$f(x_1) = x_1^2 + \frac{1}{x_1+1} - 3x_1 = 1^2 + \frac{1}{1+1} - 3 \cdot 1 = -\frac{3}{2}$

$f(x_0) = x_0^2 + \frac{1}{x_0+1} - 3x_0 = (0.5)^2 + \frac{1}{0.5+1} - 3(0.5) = -\frac{7}{12}$

Step 1: $$x_2 = 1 - \frac{(-\frac{3}{2})(1-0.5)}{(-\frac{3}{2}) - (-\frac{7}{12})} = \frac{2}{11} \approx 0.181818$$

Step 2: $$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$f(x_1) = -\frac{3}{2}$

$f(x_2) = x_2^2 + \frac{1}{x_2+1} - 3x_2 = (\frac{2}{11})^2 + \frac{1}{(\frac{2}{11})+1} - 3(\frac{2}{11}) = \frac{525}{1573}$

$$x_3 = \frac{2}{11} - \frac{\frac{525}{1573}(\frac{2}{11} - 1)}{\frac{525}{1573} - (-\frac{3}{2})} = \frac{212}{641} \approx \boxed{0.330733}$$

(c) $$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

$f(x_1) = 5x_1 - 10 = 5 \cdot 1 - 10 = -5$

$f(x_0) = 5x_0 - 10 = 5(0.5) - 10 = -\frac{15}{2}$

Step 1: $$x_2 = 1 - \frac{(-5)(1-0.5)}{(-5) - (-\frac{15}{2})} = 2$$

Step 2: $$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$f(x_1) = -5$

$f(x_2) = 5x_2 - 10 = 5 \times 2 - 10 = 0$

$$x_3 = 2 - 0 = \boxed{2}$$

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

(a) Apply Secant method (use secant_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 $r_{sl} = \frac{e_i}{e_{i-1}e_{i-2}}$ and $r_l = \frac{e_i}{e_{i-1}}$ (Modify the code to calculate and print these ratios for each iteration.)

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

Recall that Secant method requires two previous approximations, so the first ratio (testing for superlinear convergence) is not defined for the first iteration.

Solution. (a) Same as precious homework, we use WolframAlpha to find true root of this function, which are:

$$r_1 = -\frac{2}{3}$$

$$r_2 = 0.5$$

$$r_3 = -1.38129848$$

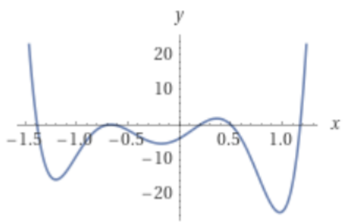
$$r_4 = 0.20518292$$

$$r_5 = 1.17611556$$

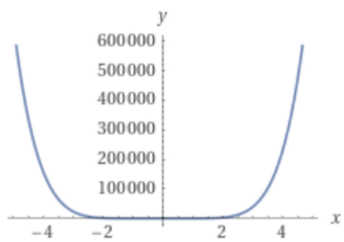
Input

$54 x^6 + 45 x^5 - 102 x^4 - 69 x^3 + 35 x^2 + 16 x - 4$

Plots



(x from −1.5 to 1.3)



(x from −5 to 5)

M Enlarge Data Customize

Alternate forms

$(3 x + 2)^2 (2 x - 1) (3 x^3 - 5 x + 1)$

$x (x (x (x (x (54 x + 45) - 102) - 69) + 35) + 16) - 4$

$(3 x + 2)^2 (6 x^4 - 3 x^3 - 10 x^2 + 7 x - 1)$

Roots

Fewer digits More digits Exact forms Step-by-step solution

$x \approx -0.666666666666667$

$x = 0.5$

$x \approx -1.38129848204399$

$x \approx 0.205182924689048$

$x \approx 1.17611555735495$

(1) Root1 $r_1 = -\frac{2}{3}$**.**

- initial guess $x_0 = -0.7$ and $x_1 = -0.6$, tolerance = 5e-7
- solution = -0.666667
- The number of iterations is 30.
- As shown in the picture.

  errors = [3.33333333e-02 6.66666667e-02 7.26885622e-02 4.21910576e-01 1.15086266e-01 2.52854671e-01 8.08372911e-02 6.26005769e-02 3.62874217e-02 2.34519192e-02 1.44487994e-02 9.02480465e-03 5.58869406e-03 3.46462614e-03 2.14388991e-03 1.32636285e-03 8.20179287e-04 5.07087648e-04 3.13465125e-04 1.93759095e-04 1.19759768e-04 7.40195133e-05 4.57480530e-05 2.82744196e-05 1.74747695e-05 1.08000827e-05 6.67484909e-06 4.12529908e-06 2.54957673e-06 1.57572788e-06 9.73860948e-07 6.01906854e-07]

  Linear error ratios (r_l): [0. 5.8043599 0.27277407 2.19708814 0.31969863 0.77440221 0.57966593 0.64628232 0.61610307 0.62460585 0.61925928 0.61993484 0.61879401 0.61867116 0.6183672 0.61826439 0.61816754 0.6181201 0.61808592 0.6180666 0.61805396 0.6180464 0.61804167 0.61803864 0.61803685 0.61803631 0.6180344 0.61803509 0.61803879 0.61806242]

  Superlinear error ratios (r_sl): [0.00000000e+00 3.75264093e+00 5.20747349e+00 2.77790424e+00 3.06263755e+00 7.17077381e+00 1.03239036e+01 1.69784195e+01 2.66334642e+01 4.28588747e+01 6.86923276e+01 1.10722470e+02 1.78567942e+02 2.88432350e+02 4.66135183e+02 7.53698067e+02 1.21896107e+03 1.97178528e+03 3.18987145e+03 5.16078117e+03 8.34977663e+03 1.35096825e+04 2.18585791e+04 3.53673820e+04 5.72251457e+04 9.25915162e+04 1.49815826e+05 2.42408388e+05 3.92239311e+05]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\M348\HW4\secant Modified.py"
Solve the problem f(x)=0 using Secant method
Enter guess 0 at root: -0.7
Enter guess 1 at root: -0.6
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess 0: x=-0.700000, error=0.033333333
Guess 1: x=-0.600000, error=0.066666667
Iter 1: x= -0.739355228845, dx= -0.139355228845, error = 0.072688562178
Iter 2: x= -0.244756091134, dx= 0.49459913771, error = 0.421910575532
Iter 3: x= -0.781752933041, dx= -0.536996841906, error = 0.115086266374
Iter 4: x= -0.919521337282, dx= -0.137768404241, error = 0.252854670615
Iter 5: x= -0.747503957809, dx= 0.172017379473, error = 0.0808372911424
Iter 6: x= -0.729267243579, dx= 0.0182367142301, error = 0.0626005769122
Iter 7: x= -0.702954088325, dx= 0.0263131552541, error = 0.0362874216581
Iter 8: x= -0.690118585872, dx= 0.012835502453, error = 0.0234519192051
Iter 9: x= -0.681115466062, dx= 0.0090031198097, error = 0.0144487993954
Iter 10: x= -0.675691471317, dx= 0.00542399474555, error = 0.00902480464987
Iter 11: x= -0.672255360724, dx= 0.00343611059301, error = 0.00558869405686
Iter 12: x= -0.67013129281, dx= 0.00212406791363, error = 0.00346462614323
Iter 13: x= -0.668810556572, dx= 0.00132073623758, error = 0.00214388990565
Iter 14: x= -0.667993029519, dx= 0.000817527053141, error = 0.00132636285251
Iter 15: x= -0.667486845954, dx= 0.0005061835654, error = 0.000820179287109
Iter 16: x= -0.667173754314, dx= 0.000313091639443, error = 0.000507087647665
Iter 17: x= -0.666980131792, dx= 0.000193622522347, error = 0.000313465125318
Iter 18: x= -0.666860425762, dx= 0.000119706030355, error = 0.000193759094963
Iter 19: x= -0.666786426435, dx= 7.39993267488e-05, error = 0.000119759768214
Iter 20: x= -0.66674068618, dx= 4.5740254922e-05, error = 7.40195132918e-05
Iter 21: x= -0.66671241472, dx= 2.82714602626e-05, error = 4.57480530291e-05
Iter 22: x= -0.666694941086, dx= 1.74736334492e-05, error = 2.827441958e-05
Iter 23: x= -0.666684141436, dx= 1.07996500859e-05, error = 1.74747694941e-05
Iter 24: x= -0.666677466749, dx= 6.67468677522e-06, error = 1.08000827189e-05
Iter 25: x= -0.666673341516, dx= 4.12523362963e-06, error = 6.67484908934e-06
```

```
Iter 26: x= -0.666670791966, dx= 2.5495500089e-06, error = 4.12529908045e-06
Iter 27: x= -0.666669216243, dx= 1.57572234975e-06, error = 2.54957673074e-06
Iter 28: x= -0.666668242395, dx= 9.73848847165e-07, error = 1.57572788362e-06
Iter 29: x= -0.666667640528, dx= 6.01866935462e-07, error = 9.73860948106e-07
Iter 30: x= -0.666667268574, dx= 3.7195409363e-07, error = 6.01906854469e-07
The root is -0.666667
The number of iterations is 30
errors = [3.33333333e-02 6.66666667e-02 7.26885622e-02 4.21910576e-01
 1.15086266e-01 2.52854671e-01 8.08372911e-02 6.26005769e-02
 3.62874217e-02 2.34519192e-02 1.44487994e-02 9.02480465e-03
 5.58869406e-03 3.46462614e-03 2.14388991e-03 1.32636285e-03
 8.20179287e-04 5.07087648e-04 3.13465125e-04 1.93759095e-04
 1.19759768e-04 7.40195133e-05 4.57480530e-05 2.82744196e-05
 1.74747695e-05 1.08000827e-05 6.67484909e-06 4.12529908e-06
 2.54957673e-06 1.57572788e-06 9.73860948e-07 6.01906854e-07]
Linear error ratios (r_l): [0.         5.8043599  0.27277407 2.19708814 0.31969863 0.77440221
 0.57966593 0.64628232 0.61610307 0.62460585 0.61925928 0.61993484
 0.61879401 0.61867116 0.6183672  0.61826439 0.61816754 0.6181201
 0.61808592 0.6180666  0.61805396 0.6180464  0.61804167 0.61803864
 0.61803685 0.61803631 0.6180344  0.61803509 0.61803879 0.61806242]
Superlinear error ratios (r_sl): [0.00000000e+00 3.75264093e+00 5.20747349e+00 2.77790424e+00
 3.06263755e+00 7.17077381e+00 1.03239036e+01 1.69784195e+01
 2.66334642e+01 4.28588747e+01 6.86923276e+01 1.10722470e+02
 1.78567942e+02 2.88432350e+02 4.66135183e+02 7.53698067e+02
 1.21896107e+03 1.97178528e+03 3.18987145e+03 5.16078117e+03
 8.34977663e+03 1.35096825e+04 2.18585791e+04 3.53673820e+04
 5.72251457e+04 9.25915162e+04 1.49815826e+05 2.42408388e+05
 3.92239311e+05]

Process finished with exit code 0
```

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\M348\HW4\secant Modified.py"
Solve the problem f(x)=0 using Secant method
Enter guess 0 at root: 0.4
Enter guess 1 at root: 0.6
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess 0: x=0.400000, error=0.1
Guess 1: x=0.600000, error=0.1
Iter 1: x= 0.459531456858, dx= -0.140468543142, error = 0.0404685431417
Iter 2: x= 0.486550989519, dx= 0.027019532661, error = 0.0134490104806
Iter 3: x= 0.502859623441, dx= 0.0163086339218, error = 0.00285962344122
Iter 4: x= 0.499833158272, dx= -0.00302646516942, error = 0.000166841728199
Iter 5: x= 0.499998036691, dx= 0.000164878419148, error = 1.96330905095e-06
Iter 6: x= 0.500000001363, dx= 1.96467219806e-06, error = 1.36314715071e-09
Iter 7: x= 0.5, dx= -1.36315826244e-09, error = 1.11022302463e-14
The root is 0.500000
The number of iterations is 7
errors = [1.00000000e-01 1.00000000e-01 4.04685431e-02 1.34490105e-02
 2.85962344e-03 1.66841728e-04 1.96330905e-06 1.36314715e-09
 1.11022302e-14]
Linear error ratios (r_l): [0.00000000e+00 3.32332460e-01 2.12627051e-01 5.83439504e-02
 1.17674941e-02 6.94311041e-04 8.14455742e-06]
Superlinear error ratios (r_sl): [0.         5.2541316  4.33815934 4.11505023 4.16149514 4.14838276]

Process finished with exit code 0
```

(2) Root2 $r_2 = 0.5$**.**

- initial guess $x_0 = 0.4$ and $x_1 = 0.6$, tolerance = 5e-7
- solution = 0.500000
- The number of iterations is 7.
- As shown in the picture.

  errors = [1.00000000e-01 1.00000000e-01 4.04685431e-02 1.34490105e-02 2.85962344e-03 1.66841728e-04 1.96330905e-06 1.36314715e-09 1.11022302e-14]

  Linear error ratios (r_l): [0.00000000e+00 3.32332460e-01 2.12627051e-01 5.83439504e-02 1.17674941e-02 6.94311041e-04 8.14455742e-06]

  Superlinear error ratios (r_sl): [0. 5.2541316 4.33815934 4.11505023 4.16149514 4.14838276]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\M348\HW4\secant Modified.py"
Solve the problem f(x)=0 using Secant method
Enter guess 0 at root: -1.5
Enter guess 1 at root: -1.2
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess 0: x=-1.500000, error=0.11870152
Guess 1: x=-1.200000, error=0.18129848
Iter 1: x= -1.28403005813, dx= -0.084030058125, error = 0.097268421875
Iter 2: x= -1.68801741129, dx= -0.403987353163, error = 0.306718931288
Iter 3: x= -1.30827030144, dx= 0.379747109848, error = 0.0730281785602
Iter 4: x= -1.32767301043, dx= -0.0194027089918, error = 0.0536254695684
Iter 5: x= -1.40730890642, dx= -0.0796358959836, error = 0.0260104264152
Iter 6: x= -1.37480149781, dx= 0.0325074086041, error = 0.00649698218894
Iter 7: x= -1.38060120375, dx= -0.00579970593492, error = 0.000697276254018
Iter 8: x= -1.38131858811, dx= -0.00071738436135, error = 2.01081073321e-05
Iter 9: x= -1.38129842092, dx= 2.01671824004e-05, error = 5.90750681706e-08
Iter 10: x= -1.38129848204, dx= -6.11137155765e-08, error = 2.03864747306e-09
The root is -1.381298
The number of iterations is 10
errors = [1.18701520e-01 1.81298480e-01 9.72684219e-02 3.06718931e-01
 7.30281786e-02 5.36254696e-02 2.60104264e-02 6.49698219e-03
 6.97276254e-04 2.01081073e-05 5.90750682e-08 2.03864747e-09]
Linear error ratios (r_l): [0.00000000e+00 3.15332484e+00 2.38094787e-01 7.34312023e-01
 4.85038670e-01 2.49783763e-01 1.07323098e-01 2.88380785e-02
 2.93787313e-03 3.45094392e-02]
Superlinear error ratios (r_sl): [   0.           2.44781176    2.39408771    6.64180155    4.65793148
    4.12615679    4.43868825    4.21335606 1716.19529508]

Process finished with exit code 0
```

(3) Root3 $r_3 = -1.38129848$**.**

- initial guess $x_0 = -1.5$ and $x_1 = -1.2$, tolerance = 5e-7
- solution $= -1.381298$
- The number of iterations is 10.
- As shown in the picture.

  errors = [1.18701520e-01 1.81298480e-01 9.72684219e-02 3.06718931e-01 7.30281786e-02 5.36254696e-02 2.60104264e-02 6.49698219e-03 6.97276254e-04 2.01081073e-05 5.90750682e-08 2.03864747e-09]

  Linear error ratios (r_l): [0.00000000e+00 3.15332484e+00 2.38094787e-01 7.34312023e-01 4.85038670e-01 2.49783763e-01 1.07323098e-01 2.88380785e-02 2.93787313e-03 3.45094392e-02]

  Superlinear error ratios (r_sl): [0. 2.44781176 2.39408771 6.64180155 4.65793148 4.12615679 4.43868825 4.21335606 1716.19529508]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\M348\HW4\secant Modified.py"
Solve the problem f(x)=0 using Secant method
Enter guess 0 at root: 0
Enter guess 1 at root: 0.4
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess 0: x=0.000000, error=0.20518292
Guess 1: x=0.400000, error=0.19481708
Iter 1: x= 0.282946262846, dx= -0.117053737154, error = 0.0777633428458
Iter 2: x= -0.0472994574607, dx= -0.330245720307, error = 0.252482377461
Iter 3: x= 0.214484562718, dx= 0.261784020179, error = 0.00930164271826
Iter 4: x= 0.205247527758, dx= -0.00923703495994, error = 6.46077583252e-05
Iter 5: x= 0.205181973245, dx= -6.55545135619e-05, error = 9.4675523668e-07
Iter 6: x= 0.205182924781, dx= 9.51536369278e-07, error = 4.78113260094e-09
Iter 7: x= 0.205182924689, dx= -9.20848496959e-11, error = 4.68904776119e-09
The root is 0.205183
The number of iterations is 7
errors = [2.05182920e-01 1.94817080e-01 7.77633428e-02 2.52482377e-01
 9.30164272e-03 6.46077583e-05 9.46755237e-07 4.78113260e-09
 4.68904776e-09]
Linear error ratios (r_l): [0.         3.24680458 0.03684076 0.00694584 0.01465389 0.00505002
 0.98073995]
Superlinear error ratios (r_sl): [0.00000000e+00 4.73754842e-01 2.75102127e-02 1.57540924e+00
 7.81642923e+01 1.03589599e+06]

Process finished with exit code 0
```

(4) **Root4** $r_4 = 0.20518292$**.**

- initial guess $x_0 = 0$ and $x_1 = 0.4$, tolerance = 5e-7
- solution = 0.20518292
- The number of iterations is 7.
- As shown in the picture.

  errors = [2.05182920e-01 1.94817080e-01 7.77633428e-02 2.52482377e-01 9.30164272e-03 6.46077583e-05 9.46755237e-07 4.78113260e-09 4.68904776e-09]

  Linear error ratios (r_l): [0. 3.24680458 0.03684076 0.00694584 0.01465389 0.00505002 0.98073995]

  Superlinear error ratios (r_sl): [0.00000000e+00 4.73754842e-01 2.75102127e-02 1.57540924e+00 7.81642923e+01 1.03589599e+06]

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\M348\HW4\secant Modified.py"
Solve the problem f(x)=0 using Secant method
Enter guess 0 at root: 1
Enter guess 1 at root: 1.3
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess 0: x=1.000000, error=0.17611556
Guess 1: x=1.300000, error=0.12388444
Iter 1: x= 1.08744893682, dx= -0.212551063183, error = 0.0886666231832
Iter 2: x= 1.13777435766, dx= 0.0503254208458, error = 0.0383412023374
Iter 3: x= 1.19586927308, dx= 0.0580949154154, error = 0.019753713078
Iter 4: x= 1.1729641998, dx= -0.0229050732807, error = 0.00315136020269
Iter 5: x= 1.17587747199, dx= 0.00291327219296, error = 0.000238088009736
Iter 6: x= 1.17611857296, dx= 0.000241100965754, error = 3.01295601846e-06
Iter 7: x= 1.17611555449, dx= -3.01846408409e-06, error = 5.50806555921e-09
Iter 8: x= 1.17611555735, dx= 2.862978338e-09, error = 2.64508726033e-09
The root is 1.176116
The number of iterations is 8
errors = [1.76115560e-01 1.23884440e-01 8.86666232e-02 3.83412023e-02
 1.97537131e-02 3.15136020e-03 2.38088010e-04 3.01295602e-06
 5.50806556e-09 2.64508726e-09]
Linear error ratios (r_l): [0.         0.43241979 0.51520849 0.15953255 0.07555087 0.0126548
 0.00182813 0.48022073]
Superlinear error ratios (r_sl): [0.00000000e+00 5.81062495e+00 4.16086454e+00 3.82464153e+00
 4.01566265e+00 7.67836555e+00 1.59385244e+05]

Process finished with exit code 0
```

(5) Root5 $r_5 = 1.17611556$**.**

- initial guess $x_0 = 1$ and $x_1 = 1.3$, tolerance = 5e-7
- solution = 1.176115562
- The number of iterations is 8.
- As shown in the picture.

  errors = [1.76115560e-01 1.23884440e-01 8.86666232e-02 3.83412023e-02 1.97537131e-02 3.15136020e-03 2.38088010e-04 3.01295602e-06 5.50806556e-09 2.64508726e-09]

  Linear error ratios (r_l): [0. 0.43241979 0.51520849 0.15953255 0.07555087 0.0126548 0.00182813 0.48022073]

  Superlinear error ratios (r_sl): [0.00000000e+00 5.81062495e+00 4.16086454e+00 3.82464153e+00 4.01566265e+00 7.67836555e+00 1.59385244e+05]

(b) The convergence type is determined by examining the linear (r_l) and superlinear (r_{sl}) error ratios for the last few iterations.

Root 1 ($r_1 = -\frac{2}{3}$). From the error ratios generated by Python, we can find that,

- Linear Error Ratios (r_l): [0.6180344 0.61803509 0.61803879 0.61806242]
- Superlinear Error Ratios (r_{sl}): [9.25915162e+04 1.49815826e+05 2.42408388e+05 3.92239311e+05]
- From the last few iterations, we can find that $r_{sl} \to \infty$ and r_l is approximately converge to 0.618, $S = 0.618 < 1$, so it is locally linear convergence.

Root 2 ($r_2 = 0.5$).

- Linear Error Ratios (r_l): [0.0117674941, 0.000694311041, 0.00000814455739]
- Superlinear Error Ratios (r_{sl}): [4.11505023 4.16149514 4.14838276]
- From the last few iterations, we can find that $r_l \to 0$ and r_{sl} is approximately converge to 4.15, $4.15 < \infty$, so it is locally superlinear convergence.

Root 3 ($r_3 = -1.38129848$).

- Linear Error Ratios (r_l): [0.02883808, 0.00293787, 0.03450944]
- Superlinear Error Ratios (r_{sl}): [4.12615679 4.43868825 4.21335606 1716.19529508]
- From the last few iterations, we can find that $r_l \to 0$ and r_{sl} is approximately converge to 4.21, $4.21 < \infty$, so it is locally superlinear convergence.

Root 4 ($r_4 = 0.20518292$).

- Linear Error Ratios (r_l): [0.00694584 0.01465389 0.00505002 0.98073995]
- Superlinear Error Ratios (r_{sl}): [1.57540924e+00 7.81642923e+01 1.03589599e+06]
- From the last few iterations, we can find that $r_l \to 0$ and r_{sl} is approximately converge to 1.58, $1.58 < \infty$, so it is locally superlinear convergence.

Root 5 ($r_5 = 1.17611556$).

- Linear Error Ratios (r_l): [0.07555087, 0.0126548, 0.00182813]
- Superlinear Error Ratios (r_{sl}): [3.82464153e+00 4.01566265e+00 7.67836555e+00 1.59385244e+05]
- From the last few iterations, we can find that $r_l \to 0$ and r_{sl} is approximately converge to 4.015, $4.015 < \infty$, so it is locally superlinear convergence.

4. Additional Problem 3

Solution. See next page.

AD3

(a) For(A), $r=4^{\frac{1}{3}}$ is a fixed point of $g_1(x)=\frac{4x-x^3+4}{4}$ since

$$g_1(r)=\frac{4r-r^3+4}{4}=\frac{4\cdot 4^{\frac{1}{3}}-4^{\frac{1}{3}\cdot 3}+4}{4}=4^{\frac{1}{3}}=r$$

Because $g_1(r)=r$, r is a fixed point root.

For(B), $r=4^{\frac{1}{3}}$ is a fixed point of $f_1(x)=4-x^3$ since

$$f_1(r)=4-4^{\frac{1}{3}\cdot 3}=0$$

Because $f_1(r)=0$, r is a fixed point root.

For(C), $r=4^{\frac{1}{3}}$ is a fixed point of $g_2(x)=\frac{3x}{4}+\frac{1}{x^2}$ since

$$g_2(r)=\frac{3r}{4}+\frac{1}{r^2}=\frac{3\cdot 4^{\frac{1}{3}}}{4}+\frac{1}{4^{2\cdot\frac{1}{3}}}=\frac{3}{4}\cdot 4^{\frac{1}{3}}+\frac{1}{4}\cdot 4^{\frac{1}{3}}=4^{\frac{1}{3}}=r$$

Because $g_2(r)=r$, r is a fixed pointroot.

For(D), $r=4^{\frac{1}{3}}$ is a fixed point of $g_3(x)=\frac{2}{\sqrt{x}}$ since

$$g_3(r)=\frac{2}{\sqrt{4^{\frac{1}{3}}}}=\frac{2}{2^{\frac{1}{3}}}=2^{\frac{2}{3}}=4^{\frac{1}{3}}=r$$

Because $g_3(r)=r$, r is a fixed point root.

For(E), $r=4^{\frac{1}{3}}$ is a fixed point of $f_2(x)=x^3-4$ since

$$f_2(r)=4^{\frac{1}{3}\cdot 3}-4=0$$

Because $f_2(r)=0$, r is a fixed point root.

For(F), $r=4^{\frac{1}{3}}$ is a fixed point of $g_4(x)=\frac{4}{x^2}$ since

$$g_4(r)=\frac{4}{r^2}=\frac{4}{4^{\frac{1}{3}\cdot 2}}=4^{\frac{1}{3}}=r$$

Because $g_4(x)=r$, r is a fixed point root.

(b) First, analyze each speed of convergence.

For (A), FPI:

$$g_1'(x) = \frac{d}{dx}\left(\frac{4x-x^3+4}{4}\right) = \frac{1}{4}(4-3x^2)$$

$$g_1'(r) = \frac{1}{4}(4-3r^2) = \frac{1}{4}(4-3\cdot 4^{2\cdot\frac{1}{3}}) = 1-3\cdot 4^{-\frac{1}{3}} \approx -0.8899,\ |g(r)| \approx 0.8899$$

Since $|g'(r)| = 0.8899 < 1$, then FPI is locally linear convergent with $S \approx 0.8899$

For (B), Bisection Method:

$f(r) = 4 - r^3 = 4-4 = 0 \Rightarrow r$ is a root.

(B) is locally convergence with rate $S = \frac{1}{2}$ and $e_{i+1} = \frac{1}{2}e_i$.

For (C), FPI:

$$g_2'(x) = \frac{d}{dx}\left(\frac{3x}{4}+\frac{1}{x^2}\right) = \frac{3}{4}+\left(-\frac{2}{x}\right) = \frac{3}{4}-\frac{2}{x^3}$$

$$g_2'(r) = \frac{3}{4}-\frac{2}{r} = \frac{3}{4}-\frac{2}{4^{\frac{1}{3}\times 3}} = \frac{1}{4} = 0.25,\ |g_2'(r)| = \frac{1}{4} = 0.25$$

since $|g_2'(r)| \approx 0.25 < 1$, then FPI is locally linear convergent with $S = \frac{1}{4}$.

For (D), FPI:

$$g_3'(x) = \frac{d}{dx}\left(\frac{2}{\sqrt{x}}\right) = -x^{-\frac{3}{2}}$$

$$g_3'(r) = -r^{-\frac{3}{2}} = -(4^{\frac{1}{3}})^{-\frac{3}{2}} = -4^{-\frac{1}{2}} = -\frac{1}{2},\ |g_3'(r)| = \frac{1}{2} = 0.5$$

since $|g_3'(r)| = 0.5 < 1$, then FPI is locally linear convergent with $S = 0.5$.

For (E), Secant Method:

$$f_2'(x) = 3x^2,\ f_2'(r) = 3r^2 = 3\cdot 4^{\frac{1}{3}\cdot 2} = 3\cdot 4^{\frac{2}{3}} \neq 0 \Rightarrow r \text{ is simple root.}$$

(E) is locally superlinearly convergent with $e_{i+1} \approx Ce_i^{\alpha} = Ce_i^{1.62}$ $(\alpha = 1.62)$

(with $C = \left|\frac{f''(r)}{2f'(r)}\right|^{0.62}$) ($e_{i+1} \approx Ce_i \cdot e_{i-1}$)

For(F), FPI =

$$g_4'(x) = \frac{d}{dx}\left(\frac{4}{x^2}\right) = -\frac{8}{x^3}$$

$$g_4'(r) = -\frac{8}{r^3} = -\frac{8}{4^{3\cdot\frac{1}{3}}} = -2 \ , \ |g_4'(r)| = 2$$

since $|g_4'(r)| = 2 > 1$, FPI does not converge.

In conclusion, we find that

linear conv (A) $S \approx 0.8899$, (B) $S = 0.5$, (C) $S = 0.25$, (D) $S = 0.5$

superlinear conv (E) $\alpha = 1.62$ ↳ S more close to 0, it converge faster.

do not converge (F)

Therefore, the rank from fastest to slowest must be:

$E > C > B = D > A$; F does not converge.

(c) Yes. Newton Method applied to $f_1(x) = 4 - x^3$ will be faster. we already showed that $r = 4^{\frac{1}{3}}$ is a simple root of f_1 so the Newton Method is locally quadratic convergent. Specifically,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{4 - x_i^3}{-3x_i^2} = x_i + \frac{4}{3x_i} - \frac{x_i^2}{3}$$

and $e_{i+1} \approx Ce_i^2$, $\alpha = 2 \rightarrow \left(C = \left|\frac{f''(r)}{2f'(r)}\right|\right)$

$(f_1'(x) = -3x^2 = -3r^2 = -3(4^{\frac{1}{3}})^2 \approx -7.56 \neq 0)$, $M = 1$

5. Additional Problem 4 You buy a \$20,000 piece of equipment for nothing down and \$4000 per year for 6 years. What interest rate are you paying? The formula relating present worth P, annual payments A, number of years n, and interest rate i is

$$A = P\frac{i(1+i)^n}{(1+i)^n - 1}$$

Report:
- the solution method used (any from Chapter 1);
- the initial guess(es) or interval and tolerance used;
- the interest rate i with 6 correct decimal places;
- the number of iterations needed;
- the backward error of the final solution.

Solution. We apply the secant method to solve this question. From the question, we know that $P = 20000$, $A = 4000$, $n = 6$. Therefore,

$$f(i) = 4000 - 20000\frac{i(1+i)^6}{(1+i)^6 - 1}$$

- initial guess $i_0 = 0.02$ and $i_1 = 0.09$, tolerance = 5e-7
- solution $i = 0.054718 = 5.47179\%$
- The number of iterations is 4.
- As shown in the picture.

  errors = [3.47180000e-02 3.52820000e-02 8.57700862e-04 2.03537256e-05 6.27813400e-08 7.49766366e-08]

  Linear error ratios (r_l): [0. 0.02373056 0.00308451 1.19425034]

  Superlinear error ratios (r_sl): [0.00000000e+00 3.59625781e+00 5.86747782e+04]
- The backward error is calculated by substituting the computed solution, in this case the interest rate $i = 0.054718$ or 5.47179%, back into the original function and taking the absolute value of the result. The function $f(i)$ is defined as:

$$f(i) = 4000 - 20000 \cdot \frac{i(1+i)^6}{(1+i)^6 - 1}$$

Therefore, the backward error is given by:

$$\text{Backward Error} = |f(i)| \approx 0.000951$$

```
C:\Users\13464\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\13464\Desktop\M348\HW4\secant Modified.py"
Solve the problem f(x)=0 using Secant method
Enter guess 0 at root: 0.02
Enter guess 1 at root: 0.09
Enter tolerance: 5e-7
Enter maxIteration: 100
Monitor iterations? (1/0): 1
Guess 0: x=0.020000, error=0.034718
Guess 1: x=0.090000, error=0.035282
Iter 1: x= 0.0538602991383, dx= -0.0361397008617, error = 0.000857700861691
Iter 2: x= 0.0546976462744, dx= 0.000837347136046, error = 2.03537256449e-05
Iter 3: x= 0.0547179372187, dx= 2.02909443049e-05, error = 6.27813400225e-08
Iter 4: x= 0.0547179250234, dx= -1.2195296535e-08, error = 7.4976636559e-08
The root is 0.054718
The number of iterations is 4
errors = [3.47180000e-02 3.52820000e-02 8.57700862e-04 2.03537256e-05
 6.27813400e-08 7.49766366e-08]
Linear error ratios (r_l): [0.         0.02373056 0.00308451 1.19425034]
Superlinear error ratios (r_sl): [0.00000000e+00 3.59625781e+00 5.86747782e+04]

Process finished with exit code 0
```

For $i = 0.054718$, the backward error was found to be approximately 0.000951, which quantifies the deviation of the computed solution from the expected result based on the original equation.

```
    #!/usr/bin/env python3
"""
 SECANT METHOD

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

 The main function is secant:

 [state, x, errors] = secant(g0, g1,tolerance, maxIteration, debug)

 Inputs:
   g0             The initial guess at the solution.
   g1              The second 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.
 Return:
   state          An error status code.
     SUCCESS      Successful termination.
     WONT_STOP    Error: Exceeded maximum number of iterations.
     BAD_ITERATE  Error: The function had a vanishing derivative.

  Remark: We assume we are given the function
   f              The name of the function for which a root is sought.
"""
from math import sqrt
from numpy import zeros
############################## VARIABLES #############################
SUCCESS = 0
```

```
WONT_STOP = 1
BAD_ITERATE = 2
x_true = 0.054718
############################## 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
        #4000 - 20000 * (x * (1 + x) ** 6 / ((1 + x) ** 6 - 1)) AD4

def secant(g0, g1,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"

    errors = zeros(MAX_ITERS+2)
    x = g1
    f0 = f(g0)
    errors[0] = abs(g0-x_true)
    errors[1] = abs(g1-x_true)
    ratios_l = zeros(MAX_ITERS)
    ratios_sl = zeros(MAX_ITERS)

    if debug:
        print("Guess 0: x=%f, error=%.8g"%(g0,errors[0]))
        print("Guess 1: x=%f, error=%.8g"%(g1,errors[1]))

    ## Secant Loop
    for itn in range(1,MAX_ITERS+1):
        fx = f(x)
        if(abs(fx-f0) < eps):
            state = BAD_ITERATE
```

```
            iters = itn
            return state,x,errors,iters,ratios_l[:itn-1], ratios_sl[:itn-2]

        dx = -f(x)*(x-g0)/(fx-f0)
        g0 = x
        f0 = fx
        x += dx
        err = abs(x - x_true)
        errors[itn+1] = err

        if itn > 1:
            ratios_l[itn - 1] = errors[itn + 1]/ errors[itn]
            if itn > 2:
                ratios_sl[itn - 2] = errors[itn+1] / (errors[itn] * errors[itn - 1])
        if debug:
            print(fmt % (itn, x, dx,err))

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

    state = WONT_STOP
    iter = itn
    return state,x, errors, MAX_ITERS, ratios_l[:MAX_ITERS], ratios_sl[:MAX_ITERS-1]


################################ MAIN ###############################
###input
print("Solve the problem f(x)=0 using Secant method")
x0 = float(input("Enter guess 0 at root: "))
x1 = float(input("Enter guess 1 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_sl] = secant(x0,x1,tol,maxIter,debug)
if s == SUCCESS:
    print(f"The root is {x:.6f}")
    print("The number of iterations is %d"%(iters))
elif s == WONT_STOP:
    print("ERROR: Failed to converge in %d iterations!"%(maxIter))
elif s == BAD_ITERATE:
    print("ERROR: Obtained a vanishing derivative!")
    exit(1)
else:
    print("ERROR: Coding error!")
    exit(1)

errors = errors[:iters+2]
print("errors =",errors)
print("Linear error ratios (r_l):", ratios_l)
print("Superlinear error ratios (r_sl):", ratios_sl)
```