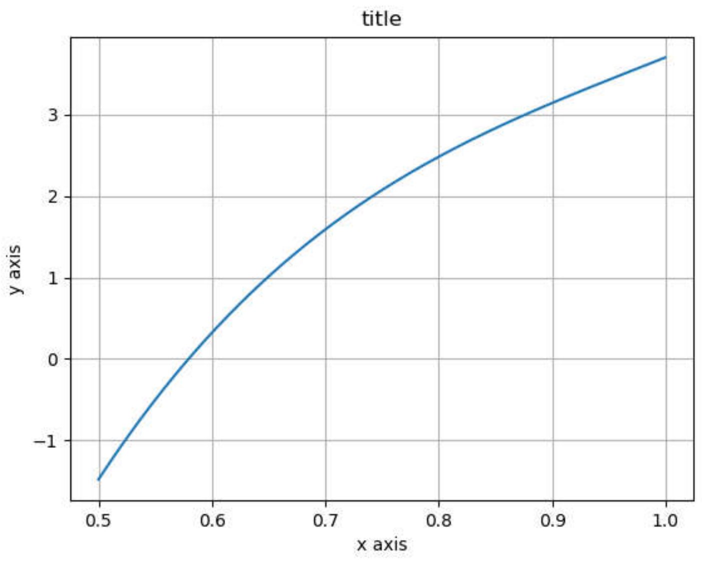
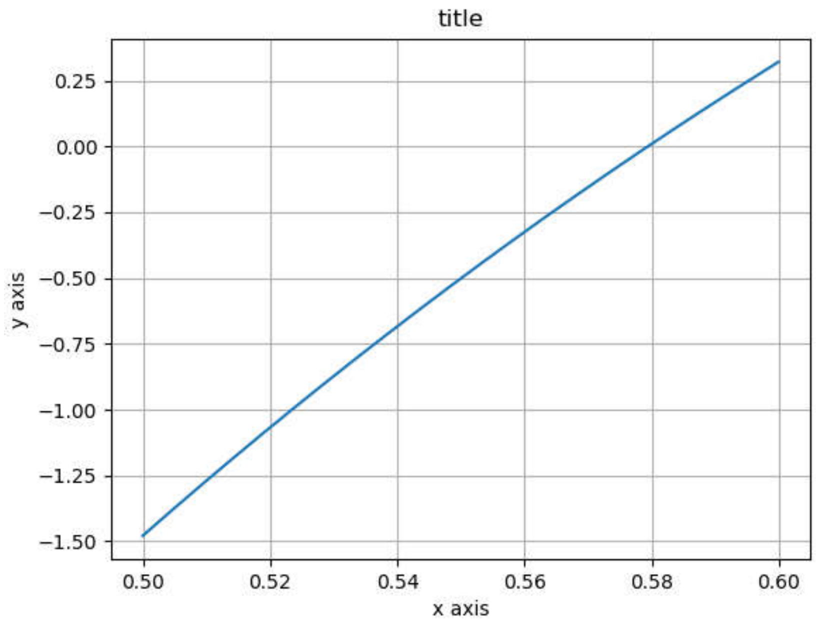
Hands-On Topic 5

Ex 5.3 Pg 142

a) The visual inspection of the function f(x) reveals the roots to be between 0.5 and 0.6.

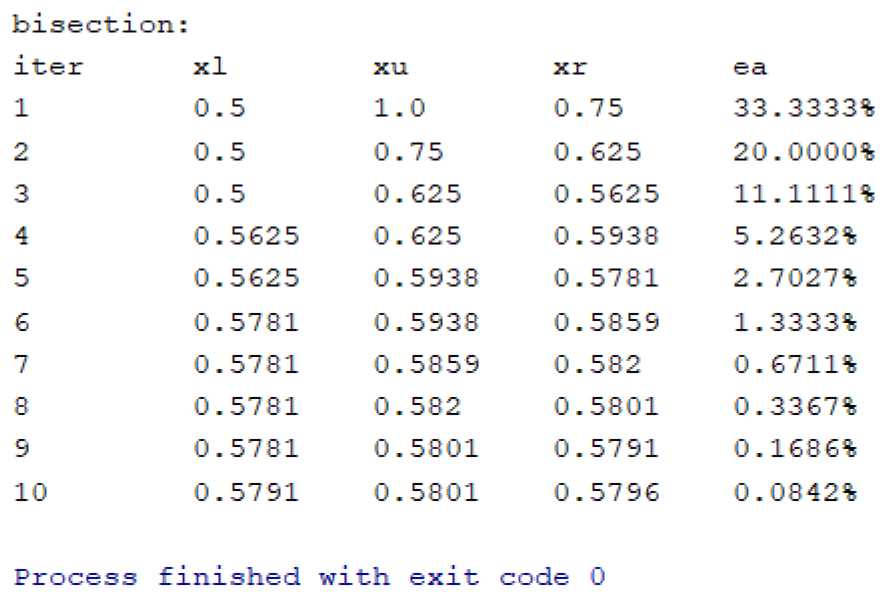


I plotted another graph with xl = 0.5 and xu = 0.6



Looks like the real root is at x = 0.58

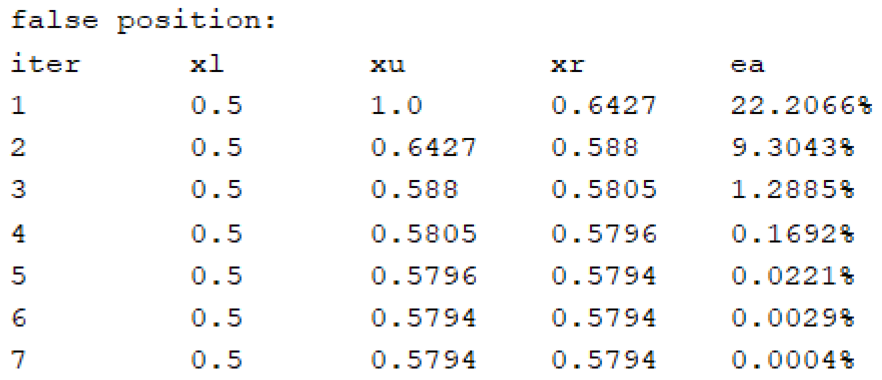
b) The bisection yields a similar result. The value of the real root comes out to be 0.5796 with es = 10%.



```
bisection:
iter     xl       xu       xr       ea
1        0.5      1.0      0.75     33.3333%
2        0.5      0.75     0.625    20.0000%
3        0.5      0.625    0.5625   11.1111%
4        0.5625   0.625    0.5938   5.2632%
5        0.5625   0.5938   0.5781   2.7027%
6        0.5781   0.5938   0.5859   1.3333%
7        0.5781   0.5859   0.582    0.6711%
8        0.5781   0.582    0.5801   0.3367%
9        0.5781   0.5801   0.5791   0.1686%
10       0.5791   0.5801   0.5796   0.0842%

Process finished with exit code 0
```

c) False position also achieves a similar result. However, the convergence of the false position method is quicker than the bisection method. This can be concluded from the fact that the es = 0.2% which is lower than es=10% in the bisection method. However it takes only 7 iterations for the false position method to converge to the real root of the function.



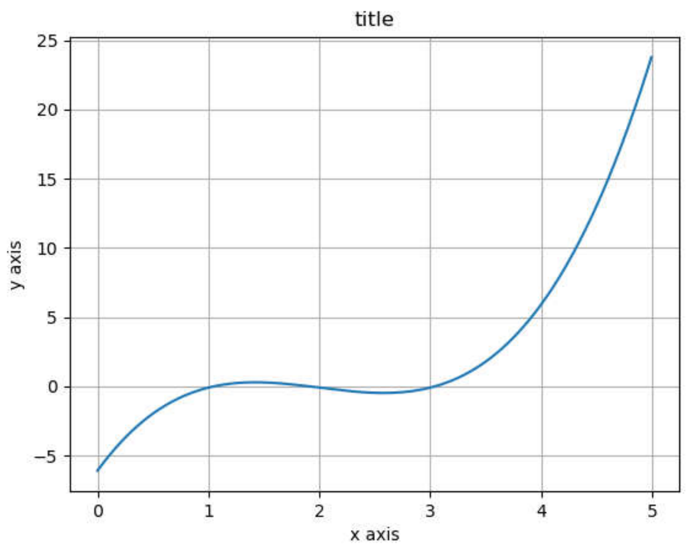
```
false position:
iter     xl       xu       xr       ea
1        0.5      1.0      0.6427   22.2066%
2        0.5      0.6427   0.588    9.3043%
3        0.5      0.588    0.5805   1.2885%
4        0.5      0.5805   0.5796   0.1692%
5        0.5      0.5796   0.5794   0.0221%
6        0.5      0.5794   0.5794   0.0029%
7        0.5      0.5794   0.5794   0.0004%
```

The real root of the function converges to x=0.5794 with the false position method.

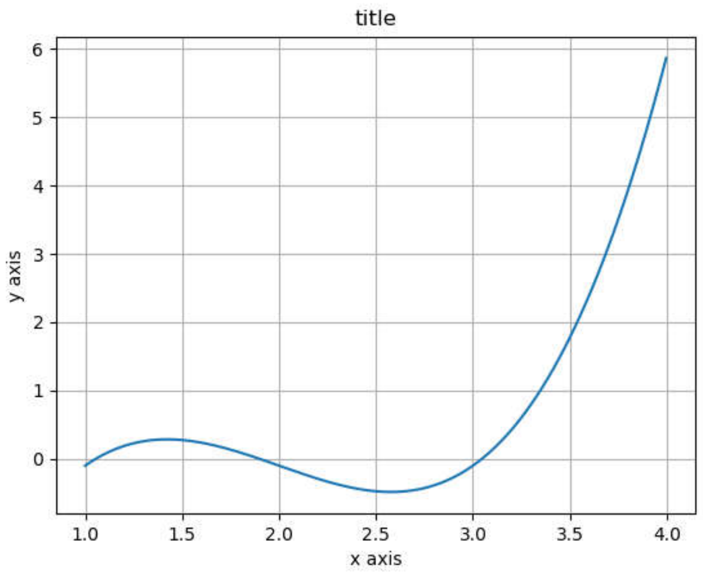
The brute force in the bisection method is highly inefficient. False position has its advantages due to its graphical insight. Bisection method does not really take into consideration about how far or close the initial guesses lie from zero. The root that is closer to zero is more likely to be the real root. However, false position finds out the real root by drawing a straight line from both the roots and finding this line’s intersection to the x axis. This pans out to be a better estimate of the real root for the function. This is evident in the results of both these methods in b) and c) above.

Ex-6.9 Pg 174

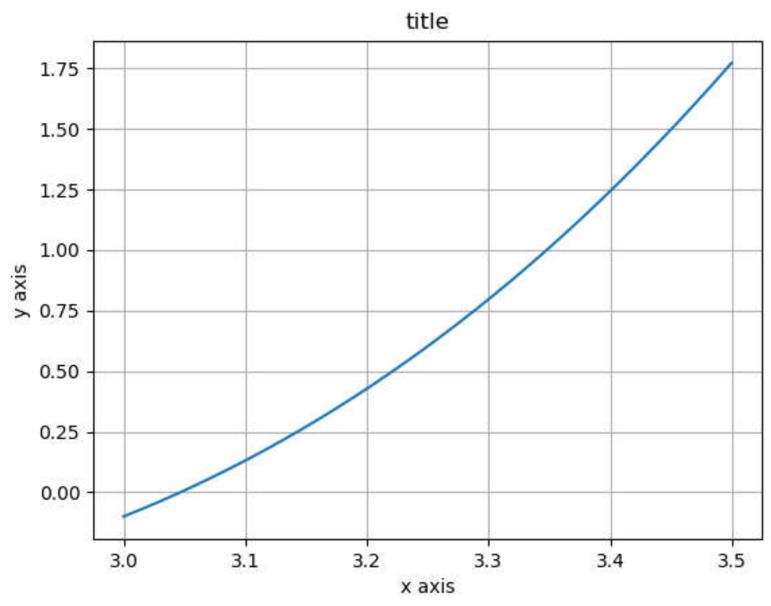
a) The visual inspection of the function f(x) reveals the roots to between x = 1 and x = 4.



I plotted another graph with x = 1 and x = 4.

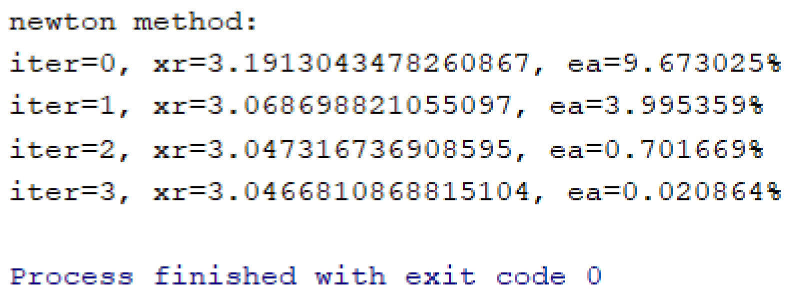


This reveals the highest real root between 3 and 3.5.



This reveals the real root to be x = 3.05.

b) The Newton-Raphson method after 3 iterations converges to a real root of x = 3.046.

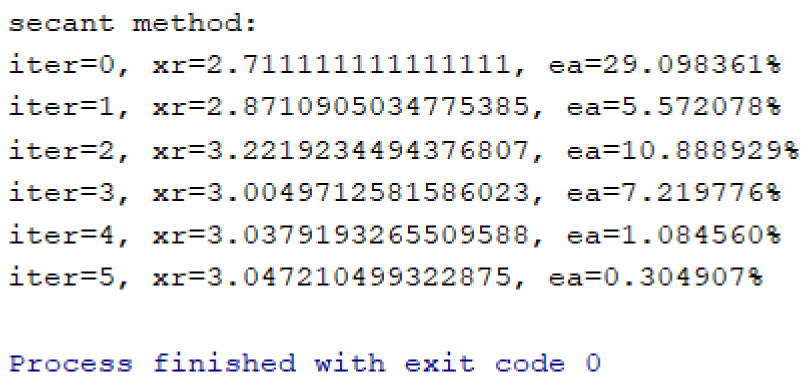


```
newton method:
iter=0, xr=3.1913043478260867, ea=9.673025%
iter=1, xr=3.068698821055097, ea=3.995359%
iter=2, xr=3.047316736908595, ea=0.701669%
iter=3, xr=3.0466810868815104, ea=0.020864%

Process finished with exit code 0
```

Newton-Raphson method is one of the most-widely used root-locating formulas. It is observed that this method rapidly converges on the true root. The approximate percent relative error decreases very quickly to the secant method shown below. This is one of the most efficient techniques to find real roots among all the different methods shown in this problem.

c) The Secant method converges after 6 iterations to a real root of x = 3.0472



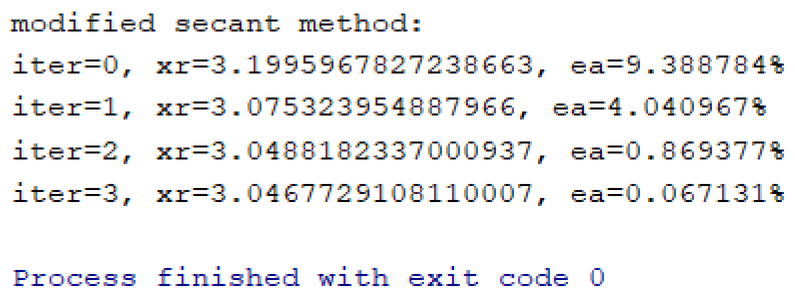
```
secant method:
iter=0, xr=2.711111111111111, ea=29.098361%
iter=1, xr=2.8710905034775385, ea=5.572078%
iter=2, xr=3.2219234494376807, ea=10.888929%
iter=3, xr=3.0049712581586023, ea=7.219776%
iter=4, xr=3.0379193265509588, ea=1.084560%
iter=5, xr=3.047210499322875, ea=0.304907%

Process finished with exit code 0
```

A major problem in the Newton-Raphson method is the evaluation of the derivative. The secant method is calculated by replacing the derivative with a backward finite divided difference. The two initial estimates were 2.5 and 3.5. It is observed that this method takes longer to converge than the Newton-Raphson method and for the approximate percent relative error to fall below 0.5%. Hence it can be concluded that this method is slightly weaker in calculating roots than the Newton-Raphson method.

d) Modified secant method

The modified secant method converges after 4 iterations to a real root of x = 3.04677.



```
modified secant method:
iter=0, xr=3.1995967827238663, ea=9.388784%
iter=1, xr=3.075323954887966, ea=4.040967%
iter=2, xr=3.0488182337000937, ea=0.869377%
iter=3, xr=3.0467729108110007, ea=0.067131%

Process finished with exit code 0
```

Modified secant method employs the use of a perturbation factor instead of using two initial estimates as done in the secant method. In this case, this proves to be more effective since the value converges to the real root within 4 iterations. Within four iterations, the approximate percent relative error falls down below 0.5%. This is almost as effective as the Newton-Raphson method and better than the secant method which employs two initial estimates to calculate the real root. If the first three iterations are considered, then the Newton Raphson method is slightly better than the modified secant since the approximate error is 0.7% instead of 0.8%. However the error in the secant method is almost at 10% which is more than 10x worse off than the remaining two. It is also important to employ the use of an appropriate perturbation factor in the modified secant method.