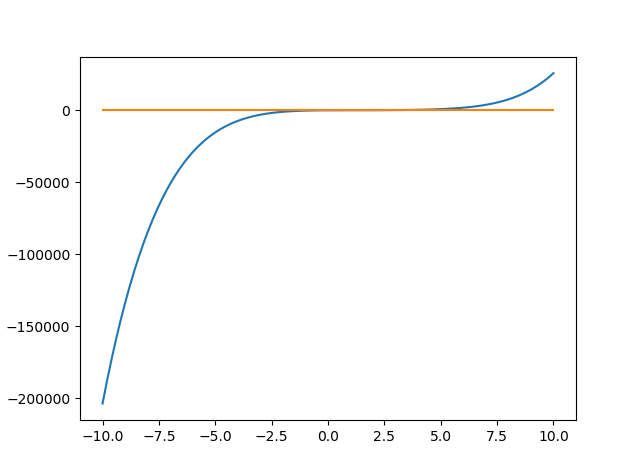
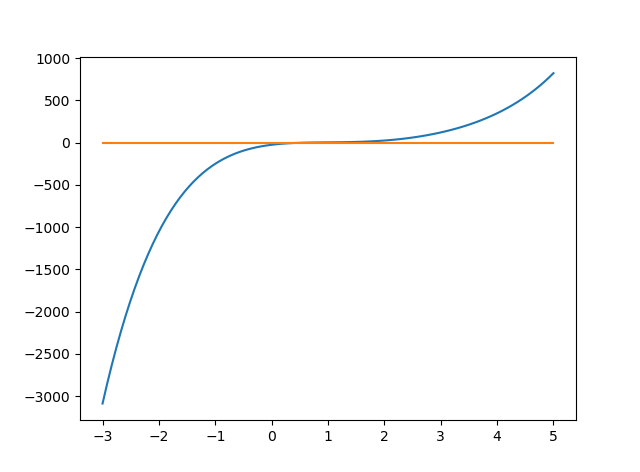
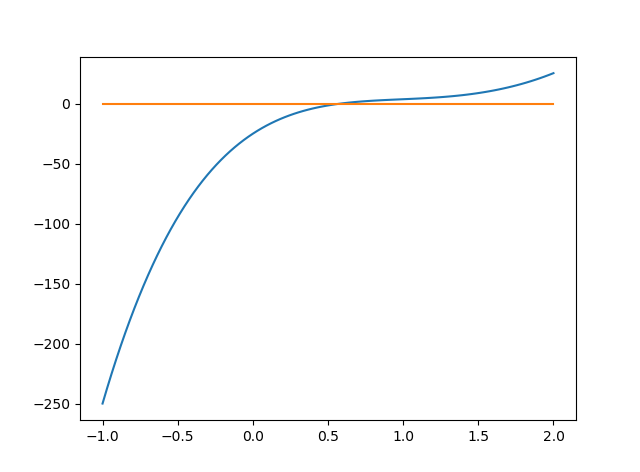
5.3  
(1) Graphic Method  
  
To Solve this problem graphically, I first plot the function from -10 to 10 and find there is a root in the range of -3 to 5 

Then I plot the function from -3 to 5 and find the root is between -1 to 2



From the plot from -1 to 2, the estimate of root is about 0.55



Overall, graphic method is simple but time consuming and accuracy is bad (root = 0.579)

(2)Bisection Method  
  
I implemented the bisect method as in the text book. It takes 4 iterations to find the root as 0.59375. The detail results are as follow

1 | 0.75

2 | 0.625

3 | 0.5625

4 | 0.59375

Overall, bisect is simple to implement, fast but accuracy is not very good. (root = 0.579)

(3) False Position

I implemented the False Position method as in the text book. It takes 12 iterations to find the root as 0.5815129348175008. The detail results are as follow

0 0.8572721786360893

1 0.736048211968592

2 0.6341448264192099

3 0.5473362069943025

4 0.5989621675165661

5 0.5658054976928671

6 0.5848919542823786

7 0.5711586996852415

8 0.5765624991594864

9 0.5820178042986787

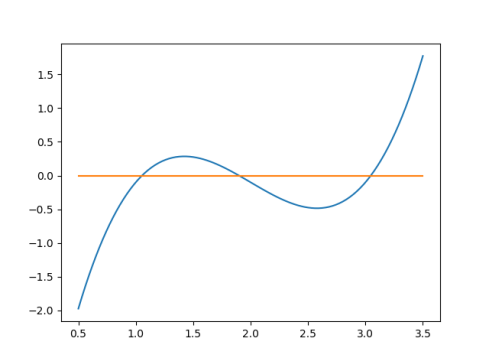
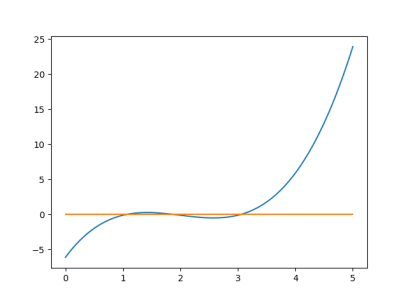
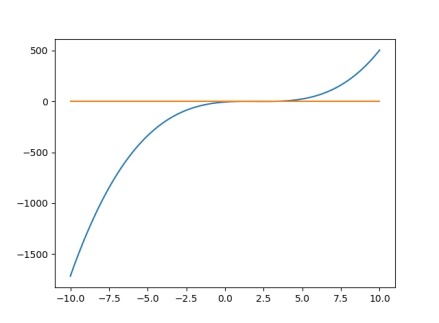
10 0.5791579980226339

11 0.5817653139535959

12 0.5815129348175008

Overall, false position is a bit difficult to implement, fast and accuracy is better. (root = 0.579)

6.9  
(1) Graphic Method  
Similar to first question, I gradually narrow down the scope and find three roots as 1.1, 1.9 and 3.1. Plots are below.



Overall, graphic method is simple but time consuming and accuracy is not very reliable (root = 1.05, 1.89, 3.04)

(2) Newton Raphson

I implemented the Newton Raphson method as in the text book. It takes 3 iterations to find the root as 3.047316736908595. The detail results are as follow

1 3.1913043478260867

2 3.068698821055097

3 3.047316736908595

Overall, Newton Raphson is simple to implement, fast but accuracy is good. (root = 3.04). But it only find one root and missed two

(3) secant

I implemented the secant method as in the text book. It takes 3 iterations to find the root as 3.2219234494376807. The detail results are as follow

1 3.5

2 2.711111111111111

3 2.8710905034775385

Overall secant is simple to implement, fast but accuracy is not good. (root = 3.04). It also only find one root and missed two

(4) modified secant

I implemented the modified secant method as in the text book. It takes 3 iterations to find the root as 3.048818233700084. The detail results are as follow

1 3.199596782723865

2 3.075323954887951

3 3.048818233700084

Overall modified secant is a bit difficult to implement, fast and accuracy is good. (root = 3.04). It also only find one root and missed two