Homework 1

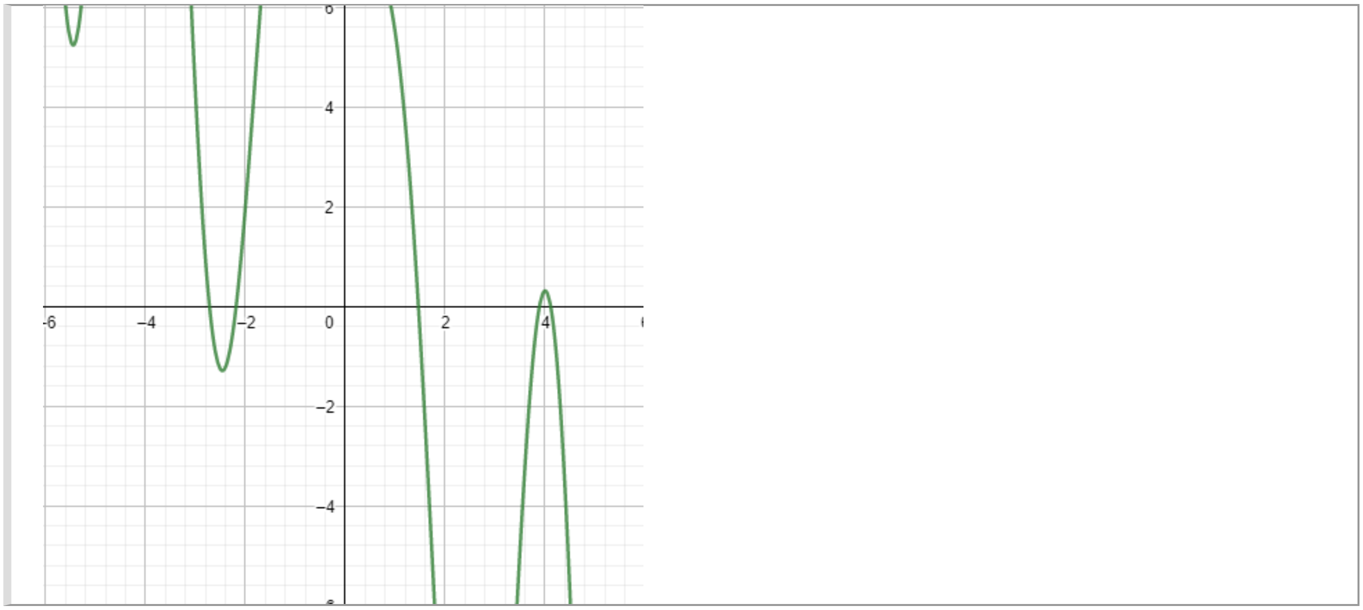
WhoAmI

- 資工三乙 李昀達 407262500
- 數值方法 HW1 解方程式 $f(x) = 0$

方程式選擇

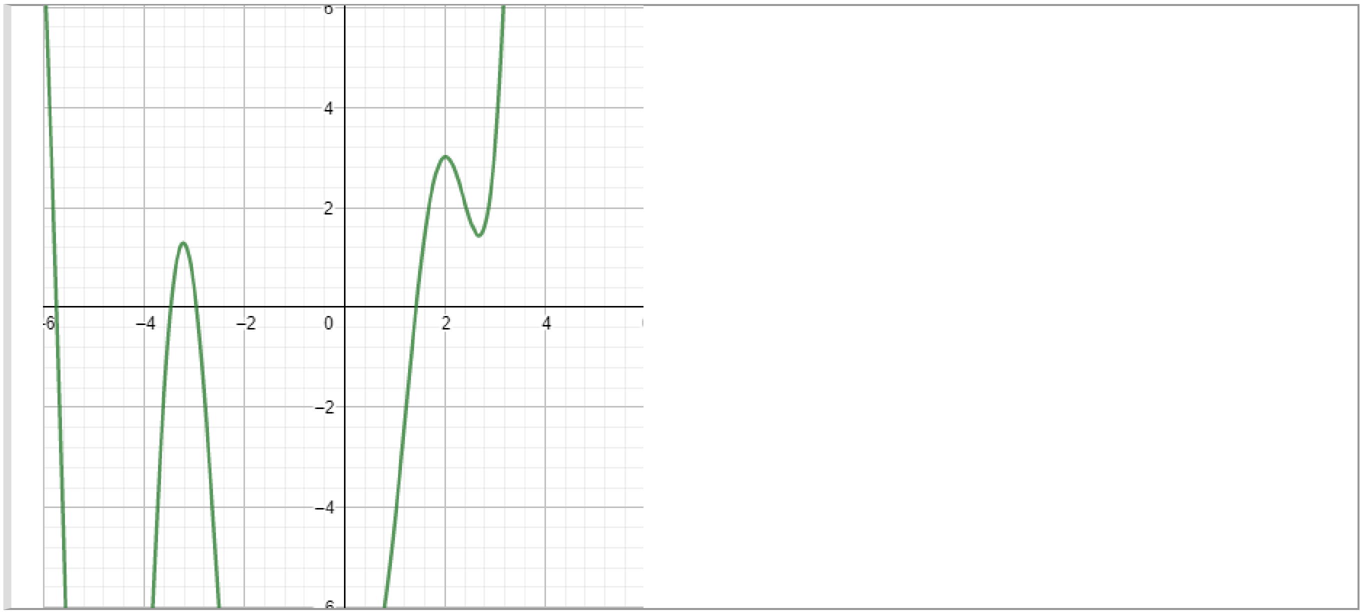
共同方程式

- $4.98\cos x + 3.2x\sin 2x - 3x + 2.9 = 0$

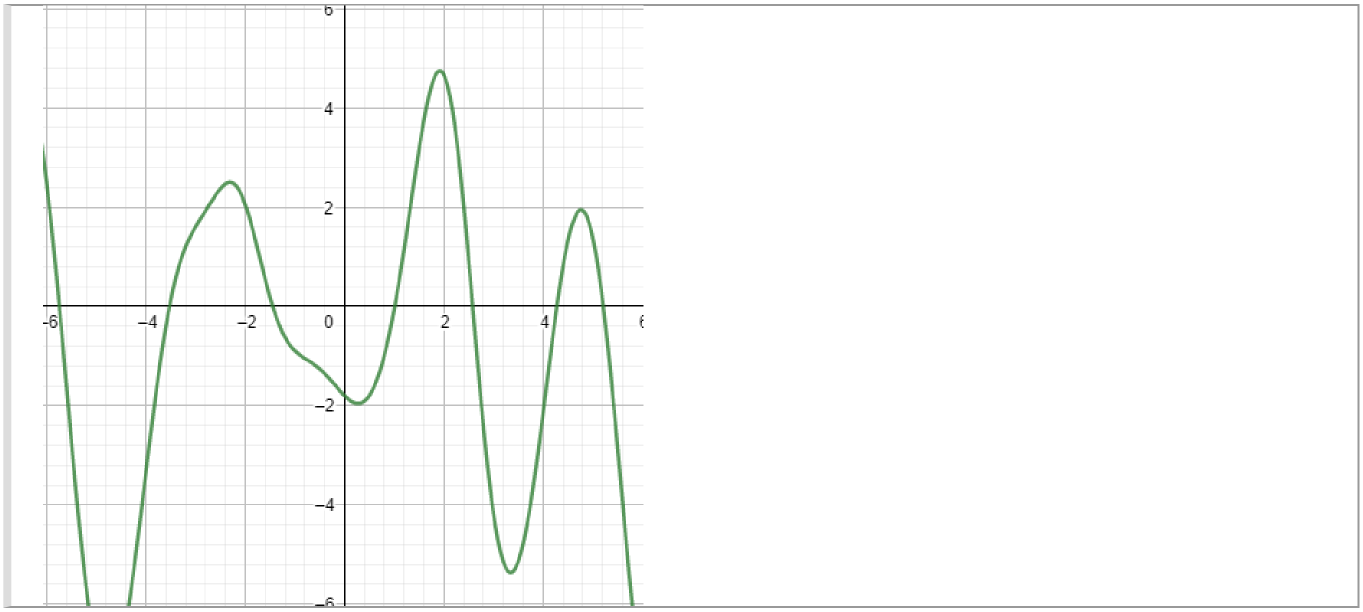


自選方程式

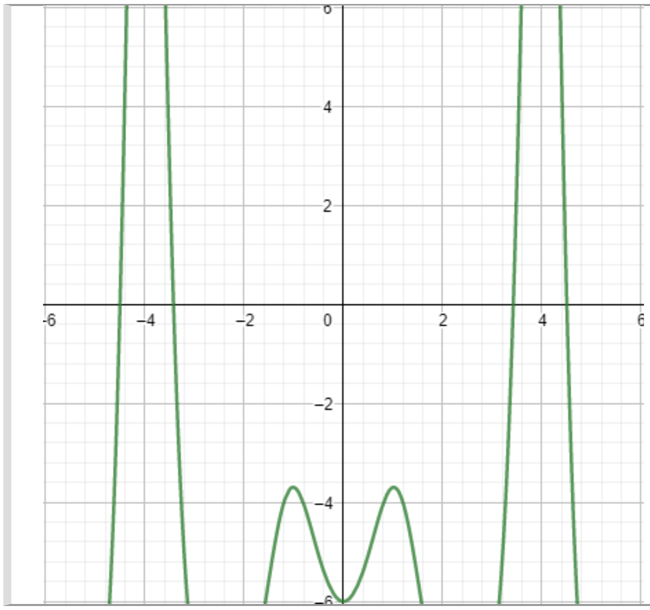
- $e^x - 3x * cos(2x) = 8.3$



- $e^{x*sin(x)} - x*cos(2*x) = 2.8$



- $4*e^{x*sin(x)*cos(x)} - 10$

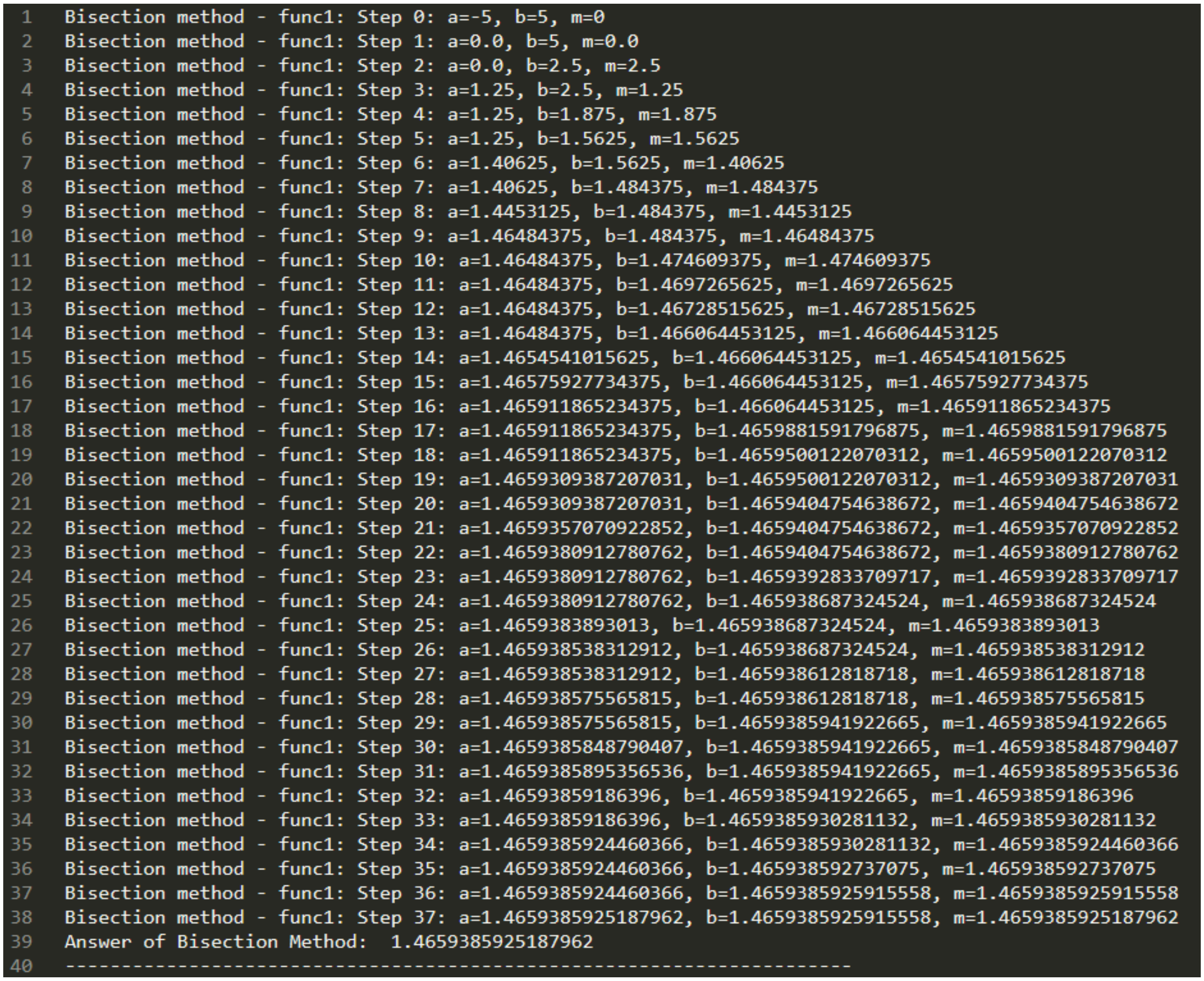


基本程式結構說明

- `main.py`
  - 主程式，主要配置要跑的所有 methods 並進行運行
- `func.py`
  - func1, func2, func3, func4 為測試運行的方程式
  - `_fixed` 為定點法所需要的方程式
- `methods.py`
  - 內部為所有 methods 的方程式

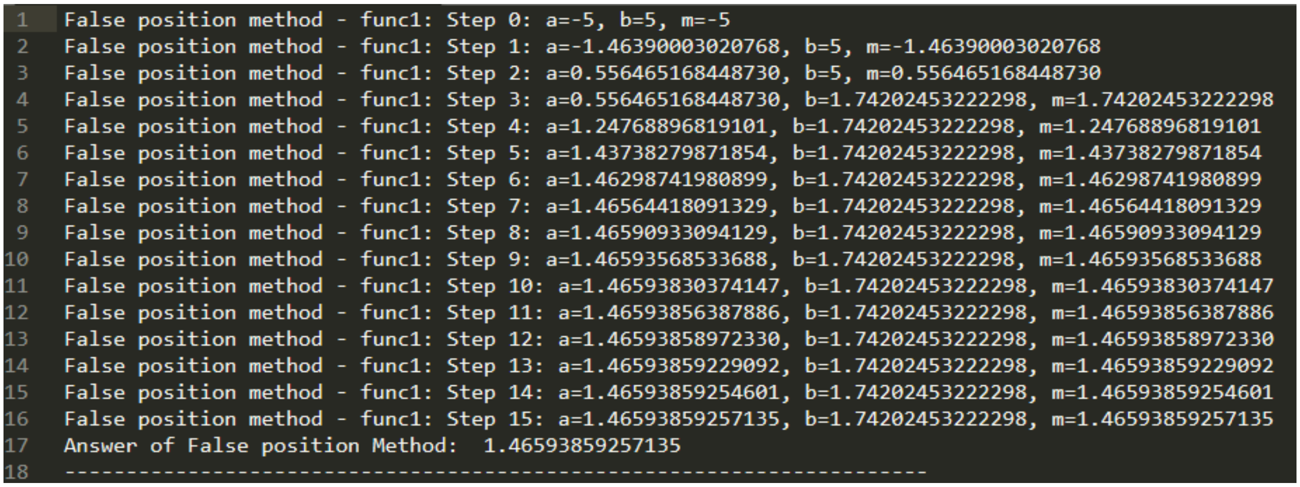
執行狀況

- 誤差值為 $eps = 10^{-10}$
- Bisection methods
  - 執行 37 步



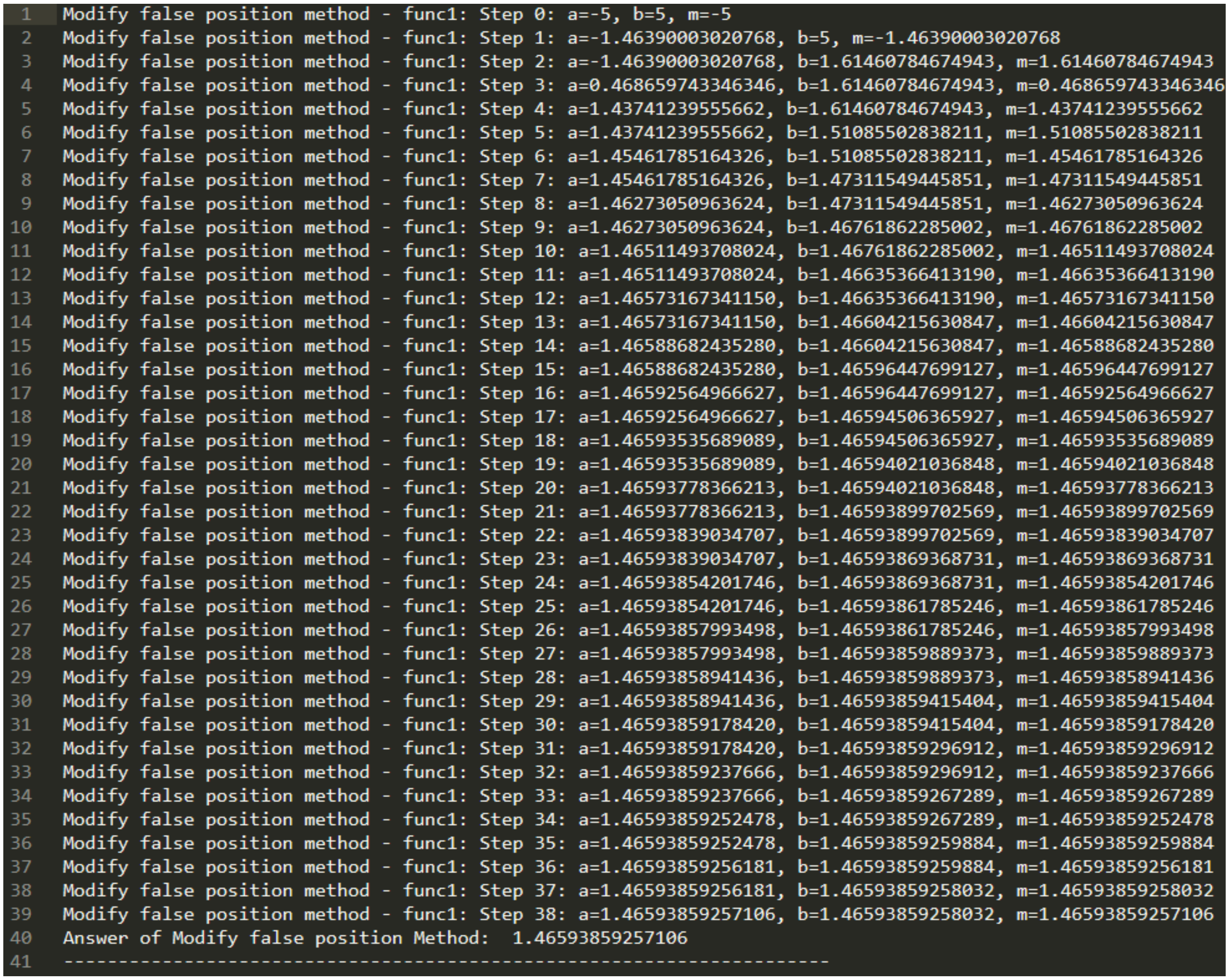
```
Bisection method - func1: Step 0: a=-5, b=5, m=0
Bisection method - func1: Step 1: a=0.0, b=5, m=0.0
Bisection method - func1: Step 2: a=0.0, b=2.5, m=2.5
Bisection method - func1: Step 3: a=1.25, b=2.5, m=1.25
Bisection method - func1: Step 4: a=1.25, b=1.875, m=1.875
Bisection method - func1: Step 5: a=1.25, b=1.5625, m=1.5625
Bisection method - func1: Step 6: a=1.40625, b=1.5625, m=1.40625
Bisection method - func1: Step 7: a=1.40625, b=1.484375, m=1.484375
Bisection method - func1: Step 8: a=1.4453125, b=1.484375, m=1.4453125
Bisection method - func1: Step 9: a=1.46484375, b=1.484375, m=1.46484375
Bisection method - func1: Step 10: a=1.46484375, b=1.474609375, m=1.474609375
Bisection method - func1: Step 11: a=1.46484375, b=1.4697265625, m=1.4697265625
Bisection method - func1: Step 12: a=1.46484375, b=1.46728515625, m=1.46728515625
Bisection method - func1: Step 13: a=1.46484375, b=1.466064453125, m=1.466064453125
Bisection method - func1: Step 14: a=1.4654541015625, b=1.466064453125, m=1.4654541015625
Bisection method - func1: Step 15: a=1.46575927734375, b=1.466064453125, m=1.46575927734375
Bisection method - func1: Step 16: a=1.465911865234375, b=1.466064453125, m=1.465911865234375
Bisection method - func1: Step 17: a=1.465911865234375, b=1.4659881591796875, m=1.4659881591796875
Bisection method - func1: Step 18: a=1.465911865234375, b=1.4659500122070312, m=1.4659500122070312
Bisection method - func1: Step 19: a=1.4659309387207031, b=1.4659500122070312, m=1.4659309387207031
Bisection method - func1: Step 20: a=1.4659309387207031, b=1.4659404754638672, m=1.4659404754638672
Bisection method - func1: Step 21: a=1.4659357070922852, b=1.4659404754638672, m=1.4659357070922852
Bisection method - func1: Step 22: a=1.4659380912780762, b=1.4659404754638672, m=1.4659380912780762
Bisection method - func1: Step 23: a=1.4659380912780762, b=1.4659392833709717, m=1.4659392833709717
Bisection method - func1: Step 24: a=1.4659380912780762, b=1.465938687324524, m=1.465938687324524
Bisection method - func1: Step 25: a=1.4659383893013, b=1.465938687324524, m=1.4659383893013
Bisection method - func1: Step 26: a=1.465938538312912, b=1.465938687324524, m=1.465938538312912
Bisection method - func1: Step 27: a=1.465938538312912, b=1.465938612818718, m=1.465938612818718
Bisection method - func1: Step 28: a=1.465938575565815, b=1.465938612818718, m=1.465938575565815
Bisection method - func1: Step 29: a=1.465938575565815, b=1.4659385941922665, m=1.4659385941922665
Bisection method - func1: Step 30: a=1.4659385848790407, b=1.4659385941922665, m=1.4659385848790407
Bisection method - func1: Step 31: a=1.4659385895356536, b=1.4659385941922665, m=1.4659385895356536
Bisection method - func1: Step 32: a=1.46593859186396, b=1.4659385941922665, m=1.46593859186396
Bisection method - func1: Step 33: a=1.46593859186396, b=1.4659385930281132, m=1.4659385930281132
Bisection method - func1: Step 34: a=1.4659385924460366, b=1.4659385930281132, m=1.4659385924460366
Bisection method - func1: Step 35: a=1.4659385924460366, b=1.465938592737075, m=1.465938592737075
Bisection method - func1: Step 36: a=1.4659385924460366, b=1.4659385925915558, m=1.4659385925915558
Bisection method - func1: Step 37: a=1.4659385925187962, b=1.4659385925915558, m=1.4659385925187962
Answer of Bisection Method:  1.4659385925187962
----------------------------------------------------------------------
```

- False position methods
  - 執行 15 步

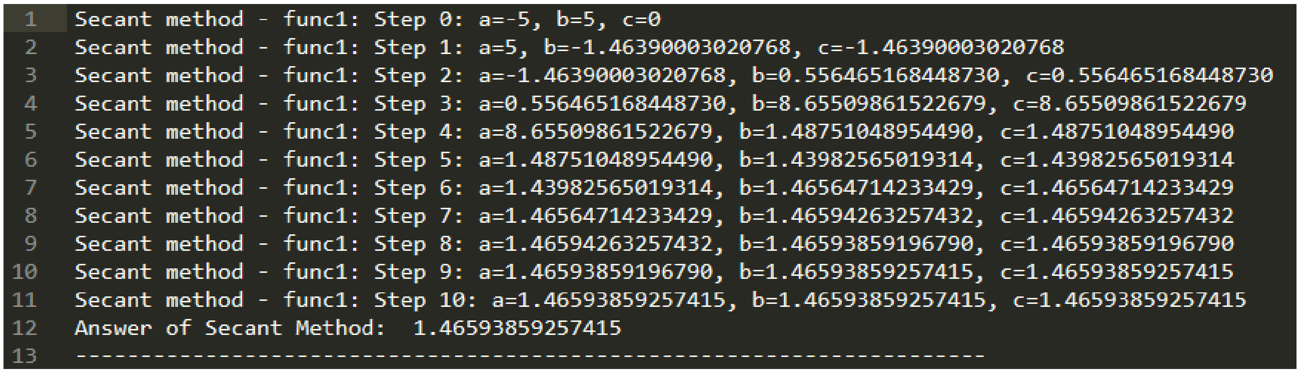


```
False position method - func1: Step 0: a=-5, b=5, m=-5
False position method - func1: Step 1: a=-1.46390003020768, b=5, m=-1.46390003020768
False position method - func1: Step 2: a=0.556465168448730, b=5, m=0.556465168448730
False position method - func1: Step 3: a=0.556465168448730, b=1.74202453222298, m=1.74202453222298
False position method - func1: Step 4: a=1.24768896819101, b=1.74202453222298, m=1.24768896819101
False position method - func1: Step 5: a=1.43738279871854, b=1.74202453222298, m=1.43738279871854
False position method - func1: Step 6: a=1.46298741980899, b=1.74202453222298, m=1.46298741980899
False position method - func1: Step 7: a=1.46564418091329, b=1.74202453222298, m=1.46564418091329
False position method - func1: Step 8: a=1.46590933094129, b=1.74202453222298, m=1.46590933094129
False position method - func1: Step 9: a=1.46593568533688, b=1.74202453222298, m=1.46593568533688
False position method - func1: Step 10: a=1.46593830374147, b=1.74202453222298, m=1.46593830374147
False position method - func1: Step 11: a=1.46593856387886, b=1.74202453222298, m=1.46593856387886
False position method - func1: Step 12: a=1.46593858972330, b=1.74202453222298, m=1.46593858972330
False position method - func1: Step 13: a=1.46593859229092, b=1.74202453222298, m=1.46593859229092
False position method - func1: Step 14: a=1.46593859254601, b=1.74202453222298, m=1.46593859254601
False position method - func1: Step 15: a=1.46593859257135, b=1.74202453222298, m=1.46593859257135
Answer of False position Method:  1.46593859257135
----------------------------------------------------------------------
```

- Modify false position methods
  - 執行 38 步

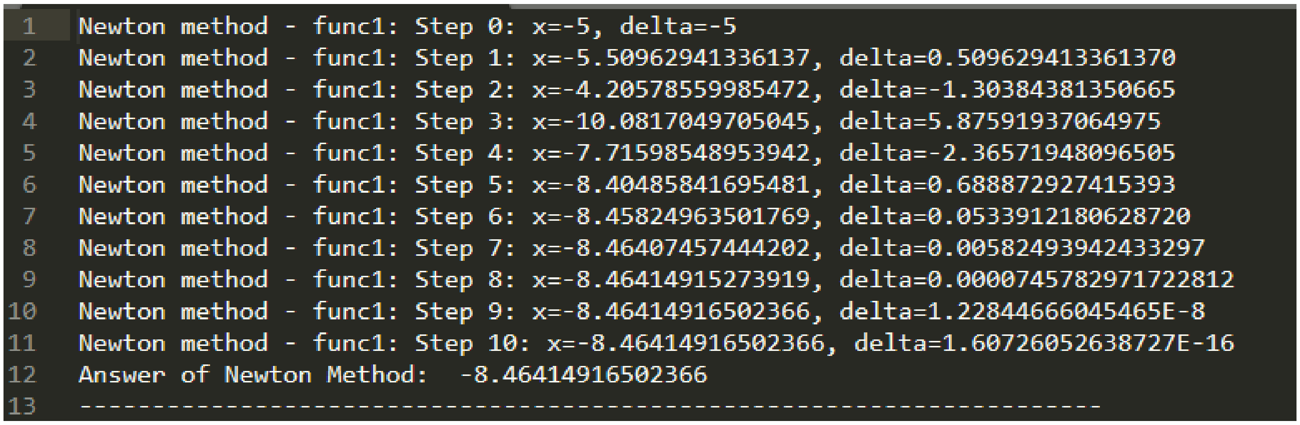
- 

```
Modify false position method - func1: Step 0: a=-5, b=5, m=-5
Modify false position method - func1: Step 1: a=-1.46390003020768, b=5, m=-1.46390003020768
Modify false position method - func1: Step 2: a=-1.46390003020768, b=1.61460784674943, m=1.61460784674943
Modify false position method - func1: Step 3: a=0.468659743346346, b=1.61460784674943, m=0.468659743346346
Modify false position method - func1: Step 4: a=1.43741239555662, b=1.61460784674943, m=1.43741239555662
Modify false position method - func1: Step 5: a=1.43741239555662, b=1.51085502838211, m=1.51085502838211
Modify false position method - func1: Step 6: a=1.45461785164326, b=1.51085502838211, m=1.45461785164326
Modify false position method - func1: Step 7: a=1.45461785164326, b=1.47311549445851, m=1.47311549445851
Modify false position method - func1: Step 8: a=1.46273050963624, b=1.47311549445851, m=1.46273050963624
Modify false position method - func1: Step 9: a=1.46273050963624, b=1.46761862285002, m=1.46761862285002
Modify false position method - func1: Step 10: a=1.46511493708024, b=1.46761862285002, m=1.46511493708024
Modify false position method - func1: Step 11: a=1.46511493708024, b=1.46635366413190, m=1.46635366413190
Modify false position method - func1: Step 12: a=1.46573167341150, b=1.46635366413190, m=1.46573167341150
Modify false position method - func1: Step 13: a=1.46573167341150, b=1.46604215630847, m=1.46604215630847
Modify false position method - func1: Step 14: a=1.46588682435280, b=1.46604215630847, m=1.46588682435280
Modify false position method - func1: Step 15: a=1.46588682435280, b=1.46596447699127, m=1.46596447699127
Modify false position method - func1: Step 16: a=1.46592564966627, b=1.46596447699127, m=1.46592564966627
Modify false position method - func1: Step 17: a=1.46592564966627, b=1.46594506365927, m=1.46594506365927
Modify false position method - func1: Step 18: a=1.46593535689089, b=1.46594506365927, m=1.46593535689089
Modify false position method - func1: Step 19: a=1.46593535689089, b=1.46594021036848, m=1.46594021036848
Modify false position method - func1: Step 20: a=1.46593778366213, b=1.46594021036848, m=1.46593778366213
Modify false position method - func1: Step 21: a=1.46593778366213, b=1.46593899702569, m=1.46593899702569
Modify false position method - func1: Step 22: a=1.46593839034707, b=1.46593899702569, m=1.46593839034707
Modify false position method - func1: Step 23: a=1.46593839034707, b=1.46593869368731, m=1.46593869368731
Modify false position method - func1: Step 24: a=1.46593854201746, b=1.46593869368731, m=1.46593854201746
Modify false position method - func1: Step 25: a=1.46593854201746, b=1.46593861785246, m=1.46593861785246
Modify false position method - func1: Step 26: a=1.46593857993498, b=1.46593861785246, m=1.46593857993498
Modify false position method - func1: Step 27: a=1.46593857993498, b=1.46593859889373, m=1.46593859889373
Modify false position method - func1: Step 28: a=1.46593858941436, b=1.46593859889373, m=1.46593858941436
Modify false position method - func1: Step 29: a=1.46593858941436, b=1.46593859415404, m=1.46593859415404
Modify false position method - func1: Step 30: a=1.46593859178420, b=1.46593859415404, m=1.46593859178420
Modify false position method - func1: Step 31: a=1.46593859178420, b=1.46593859296912, m=1.46593859296912
Modify false position method - func1: Step 32: a=1.46593859237666, b=1.46593859296912, m=1.46593859237666
Modify false position method - func1: Step 33: a=1.46593859237666, b=1.46593859267289, m=1.46593859267289
Modify false position method - func1: Step 34: a=1.46593859252478, b=1.46593859267289, m=1.46593859252478
Modify false position method - func1: Step 35: a=1.46593859252478, b=1.46593859259884, m=1.46593859259884
Modify false position method - func1: Step 36: a=1.46593859256181, b=1.46593859259884, m=1.46593859256181
Modify false position method - func1: Step 37: a=1.46593859256181, b=1.46593859258032, m=1.46593859258032
Modify false position method - func1: Step 38: a=1.46593859257106, b=1.46593859258032, m=1.46593859257106
Answer of Modify false position Method:  1.46593859257106
-----------------------------------------------------------------
```

- Secant methods
  - 執行 10 步
  - 

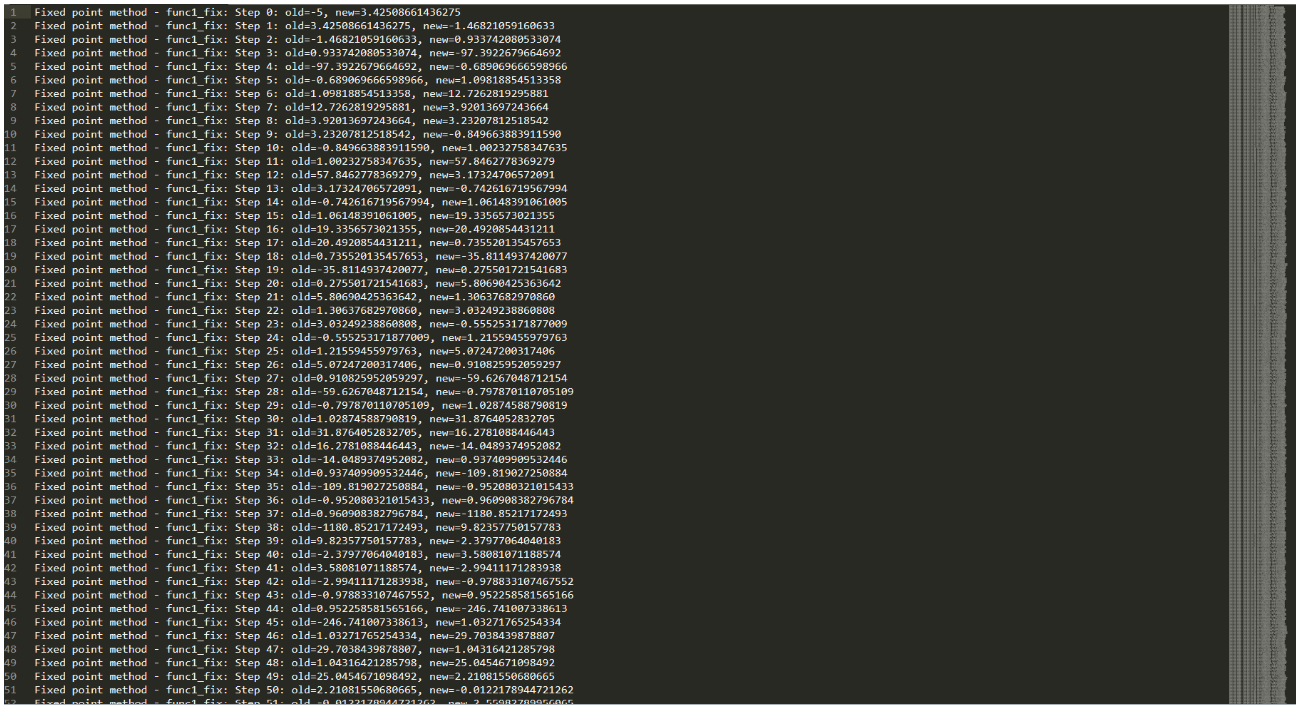
```
Secant method - func1: Step 0: a=-5, b=5, c=0
Secant method - func1: Step 1: a=5, b=-1.46390003020768, c=-1.46390003020768
Secant method - func1: Step 2: a=-1.46390003020768, b=0.556465168448730, c=0.556465168448730
Secant method - func1: Step 3: a=0.556465168448730, b=8.65509861522679, c=8.65509861522679
Secant method - func1: Step 4: a=8.65509861522679, b=1.48751048954490, c=1.48751048954490
Secant method - func1: Step 5: a=1.48751048954490, b=1.43982565019314, c=1.43982565019314
Secant method - func1: Step 6: a=1.43982565019314, b=1.46564714233429, c=1.46564714233429
Secant method - func1: Step 7: a=1.46564714233429, b=1.46594263257432, c=1.46594263257432
Secant method - func1: Step 8: a=1.46594263257432, b=1.46593859196790, c=1.46593859196790
Secant method - func1: Step 9: a=1.46593859196790, b=1.46593859257415, c=1.46593859257415
Secant method - func1: Step 10: a=1.46593859257415, b=1.46593859257415, c=1.46593859257415
Answer of Secant Method:  1.46593859257415
-----------------------------------------------------------------
```

- Newton methods
  - 執行 10 步



```
Newton method - func1: Step 0: x=-5, delta=-5
Newton method - func1: Step 1: x=-5.50962941336137, delta=0.509629413361370
Newton method - func1: Step 2: x=-4.20578559985472, delta=-1.30384381350665
Newton method - func1: Step 3: x=-10.0817049705045, delta=5.87591937064975
Newton method - func1: Step 4: x=-7.71598548953942, delta=-2.36571948096505
Newton method - func1: Step 5: x=-8.40485841695481, delta=0.688872927415393
Newton method - func1: Step 6: x=-8.45824963501769, delta=0.0533912180628720
Newton method - func1: Step 7: x=-8.46407457444202, delta=0.00582493942433297
Newton method - func1: Step 8: x=-8.46414915273919, delta=0.0000745782971722812
Newton method - func1: Step 9: x=-8.46414916502366, delta=1.22844666045465E-8
Newton method - func1: Step 10: x=-8.46414916502366, delta=1.60726052638727E-16
Answer of Newton Method:  -8.46414916502366
----------------------------------------------------------------------
```

- Fixed point methods
    - 無法收斂



```
Fixed point method - func1_fix: Step 0: old=-5, new=3.42508661436275
Fixed point method - func1_fix: Step 1: old=3.42508661436275, new=-1.46821059160633
Fixed point method - func1_fix: Step 2: old=-1.46821059160633, new=0.933742080533074
Fixed point method - func1_fix: Step 3: old=0.933742080533074, new=-97.3922679664692
Fixed point method - func1_fix: Step 4: old=-97.3922679664692, new=-0.689069666598966
Fixed point method - func1_fix: Step 5: old=-0.689069666598966, new=1.09818854513358
Fixed point method - func1_fix: Step 6: old=1.09818854513358, new=12.7262819295881
Fixed point method - func1_fix: Step 7: old=12.7262819295881, new=3.92013697243664
Fixed point method - func1_fix: Step 8: old=3.92013697243664, new=3.23207812518542
Fixed point method - func1_fix: Step 9: old=3.23207812518542, new=-0.849663883911590
Fixed point method - func1_fix: Step 10: old=-0.849663883911590, new=1.00232758347635
Fixed point method - func1_fix: Step 11: old=1.00232758347635, new=57.8462778369279
Fixed point method - func1_fix: Step 12: old=57.8462778369279, new=3.17324706572091
Fixed point method - func1_fix: Step 13: old=3.17324706572091, new=-0.742616719567994
Fixed point method - func1_fix: Step 14: old=-0.742616719567994, new=1.06148391061005
Fixed point method - func1_fix: Step 15: old=1.06148391061005, new=19.3356573021355
Fixed point method - func1_fix: Step 16: old=19.3356573021355, new=20.4920854431211
Fixed point method - func1_fix: Step 17: old=20.4920854431211, new=0.735520135457653
Fixed point method - func1_fix: Step 18: old=0.735520135457653, new=-35.8114937420077
Fixed point method - func1_fix: Step 19: old=-35.8114937420077, new=0.275501721541683
Fixed point method - func1_fix: Step 20: old=0.275501721541683, new=5.80690425363642
Fixed point method - func1_fix: Step 21: old=5.80690425363642, new=1.30637682970860
Fixed point method - func1_fix: Step 22: old=1.30637682970860, new=3.03249238860808
Fixed point method - func1_fix: Step 23: old=3.03249238860808, new=-0.555253171877009
Fixed point method - func1_fix: Step 24: old=-0.555253171877009, new=1.21559455979763
Fixed point method - func1_fix: Step 25: old=1.21559455979763, new=5.07247200317406
Fixed point method - func1_fix: Step 26: old=5.07247200317406, new=0.910825952059297
Fixed point method - func1_fix: Step 27: old=0.910825952059297, new=-59.6267048712154
Fixed point method - func1_fix: Step 28: old=-59.6267048712154, new=-0.797870110705109
Fixed point method - func1_fix: Step 29: old=-0.797870110705109, new=1.02874588790819
Fixed point method - func1_fix: Step 30: old=1.02874588790819, new=31.8764052832705
Fixed point method - func1_fix: Step 31: old=31.8764052832705, new=16.2781088446443
Fixed point method - func1_fix: Step 32: old=16.2781088446443, new=-14.0489374952082
Fixed point method - func1_fix: Step 33: old=-14.0489374952082, new=0.937409909532446
Fixed point method - func1_fix: Step 34: old=0.937409909532446, new=-109.819027250884
Fixed point method - func1_fix: Step 35: old=-109.819027250884, new=-0.952080321015433
Fixed point method - func1_fix: Step 36: old=-0.952080321015433, new=0.960908382796784
Fixed point method - func1_fix: Step 37: old=0.960908382796784, new=-1180.85217172493
Fixed point method - func1_fix: Step 38: old=-1180.85217172493, new=9.82357750157783
Fixed point method - func1_fix: Step 39: old=9.82357750157783, new=-2.37977064040183
Fixed point method - func1_fix: Step 40: old=-2.37977064040183, new=3.58081071188574
Fixed point method - func1_fix: Step 41: old=3.58081071188574, new=-2.99411171283938
Fixed point method - func1_fix: Step 42: old=-2.99411171283938, new=-0.978833107467552
Fixed point method - func1_fix: Step 43: old=-0.978833107467552, new=0.952258581565166
Fixed point method - func1_fix: Step 44: old=0.952258581565166, new=-246.741007338613
Fixed point method - func1_fix: Step 45: old=-246.741007338613, new=1.03271765254334
Fixed point method - func1_fix: Step 46: old=1.03271765254334, new=29.7038439878807
Fixed point method - func1_fix: Step 47: old=29.7038439878807, new=1.04316421285798
Fixed point method - func1_fix: Step 48: old=1.04316421285798, new=25.0454671098492
Fixed point method - func1_fix: Step 49: old=25.0454671098492, new=2.21081550680665
Fixed point method - func1_fix: Step 50: old=2.21081550680665, new=-0.0122178944721262
Fixed point method - func1_fix: Step 51: old=-0.0122178944721262, new=2.55982789956065
```

小結論

- 在上述執行狀況的結果中，
    - Bisection method 很穩定，且次數都約在 30~40 步左右
        - 優點：這個解法想法很直觀與簡單，也不會發生分母為0的問題，但是有條件限制
        - 缺點：因為每次都必須取區間中點當作新的端點，就算解在旁邊也只能慢慢找中點去逼近
    - False position method 很快速，且次數都約在 5~20 步左右
        - 優點：計算步驟與速度能比二分法還要快，不需要微分，並且會收斂
        - 缺點：有可能會遇到分母為0的問題，導致無法計算結果
    - Modify false position method 計算 range 落差較大，約 10~40 步左右
        - 與false_postion算法類似，優缺點也差不多
        - 理論上要更快，但是發現誤差較大

    - Secant method 平均計算次數從7~20次不等
        - 優點：跟牛頓法比較起來較不容易產生錯誤
        - 缺點：分母為0可能導致結果算不出來，以及找的區間內可能會無解
    - Newten method 平均計算次數從5~30次不等
        - 優點：能夠快速地找出解是多少
        - 缺點：當切線斜率為0或者微分結果趨近於零，都有可能導致算不出結果
    - Fixed point method 基本無法收斂
        - 優點：不需要微分，找一個端點就能算出結果
        - 缺點：有很高機率算不出解，因為方程式有可能會發散

感想

- 這次練習可以發現，各種算法中，其實都是為了加速跟更加精準去計算而發展，但是 fix point method 跟 newton method 的計算法我還是不是很理解，他們如何去取得正確的答案，但是確實爭取得更快速的計算跟推算結果

Reference

- 函數圖形畫法 (https://www.geogebra.org/graphing)