# Homework #2

Implement Fibonacci search and Goldensection search & Disscusion

GIST AI

20231126 황산하

*코드는 Python으로 작성했고 Report 이외에 스크립트도 Report와 함께 Zip으로 압축하여 제출하였습니다.

# INDEX

# 1. Compative Study : Fibonacci and Golden section search

사용한 함수로는 과제 1에서 사용했던 세가지 함수와 좀 더 복잡한 함수를 추가해서 총 4가지 경우에 대해서 실험을 진행했다. **1) 인수분해를 통해 간단하게 구할 수 있는 함수 (F1)**와 **2) 해를 구하는게 좀 더 복잡한 임의의 함수 (F2)**, **3) 지수함수와 같은 다른 복합적인 함수로 표현된 함수 (F3)**, 마지막으로 **4) 복잡해서 손으로는 해를 구하는 것이 거의 불가능한 함수 (F4)**. 사용한 함수는 다음과 같다.

$$\mathrm{F}_1(x) = x^3 - 3x^2 + 3x + 1$$

$$\mathrm{F}_2(x) = x^5 - 4x^4 + 3x^3 + 9x - 5$$

$$\mathrm{F}_3(x) = e^x - x^e + 3x^3 - 2x - 5$$

$$\mathrm{F}_4(x) = -20e^{-0.2\sqrt{0.5x^2}} - \mathrm{e}^{0.5\cos(2\pi x)+1} + 20 + e$$

F4 함수는 Ackley function으로 알려진 함수인데, 살짝 변형해서 y를 0으로 고정하고 unimodal 함수로 만들어서 해를 탐색할 수 있도록 만들었다. 이번에도 확실한 비교를 위해서 파라미터는 최대한 고정을 하고 과제를 진행했다. Fibonacci Search의 N은 15로 고정했다.

[표 1 Fibonacci Search 결과표]

| Fibonacci Search (local minimum) | Interval (a,b) | Time | Convergence |
|---|---|---|---|
| F1 (x=1) | -5, -4.9797 | 0.00099 | NO |
| F2 (x=0.536) | -5, -4.9797 | 0.001 | No |
| F3 (x=1.249) | -5, -4.9797 | 0.0019 | No |
| F4 (x=0) | -0.0030, 0.0009 | 0.0019 | Yes |

## 1) Fibonacci Search

Fibonacci search는 이전 값과 그 이전의 값이 다음 수열의 값이 되는 Fibonacci 수열에서 영감을 받아 만들어진 탐색 방법이다. 이 수열은 값이 점점 커지는 특성 때문에 만약 현재 값과 그 이전의 값의 비율로 범위를 줄여가며 탐색을 하게 된다면 이분법보다는 확실히 느리겠지만, 조금 더 정밀하게 탐색을 할 수 있다는 장점이 있다. 실제 결과를 보면 그렇게 느리다고 생각이 들지도 않을 정도로 빠르게 탐색한다.

하지만 F1~F3의 결과에서 볼 수 있듯이 Fibonacci Search는 단봉함수의 최솟값을 찾는 데에 최적화되어 있는 함수이기 때문에 위로 볼록하거나, F1같이 볼록하지 않은 함수의 모양의 경우에는 value를 찾지 못한다는 단점도 명확하게 존재한다. 이와 같은 경우에는 초기 범위가 애초에 local minimum 근처일 경우에 해소가 가능했다. F4에서는 local minimum이 있는 범위로 잘 수렴하는 것을 확인할 수 있었다.

[표 2 Golden Section Search 결과표]

| Golden section (Global minimum) | Interval (a,b) | Time | Convergence |
|---|---|---|---|
| F1 (x=1) | 2.236, 2.2358 | 0.000997 | Yes (iter : 15) |
| F2 (x=0.536) | 2.236, 2.23589 | 0.001996 | Yes (iter : 15) |
| F3 (x=1.249) | -5, -4.9999 | 0.006 | No (iter : 55) |
| F4 (x=0) | -2.23589, -2.23588 | 0.002 | Yes (iter : 15) |

## 2) Golden Section Search

Golden Section Serach 같은 경우, 말 그대로 황금비를 이용하여 interval을 탐색하는 알고리즘이다. 구현한 코드를 보면 사실상 Fibonacci Search랑 거의 비슷한데, interval이 줄어드는 정도가 다르다는 차이점이 있다. 실제로 구현해보고 느낀 점은 Fibonacci Search에 비해 좀 더 유연한 사용이 가능하다는 점이 있다. 위의 실험 결과를 보면 알겠지만 Fibonacci Search에 비해 어떤 local minimum으로 수렴을 했다. 전역해를 알고있는 상황에서 결과가 다소 아쉽다고 느낄 수도 있지만 Fibonacci Search는 계속해서 값을 줄여가며 Minimum value를 찾는 거에 비해 조금 더 다양한 상황에 적용할 수 있다고 결론 내릴 수 있다. F4의 결과를 보면 알 수 있듯이 Golden Section Search는 어떤 한 점으로 수렴할 가능성이 상대적으로 높아 보이는 데에 반해, 확실하게 Mimimum을 찾는 문제라면 Fibonacci Search가 더 유리할 수 있다. 왜냐하면 Fibonacci Search는 정확하게 전역해가 존재하는 지점으로 수렴했기 때문이다.

그러므로 확실하게 함수 모양 혹은 문제를 알고 있는 경우, 그것이 local minimum을 찾는 거라면 Fibonacci Search가 유리할 수 있지만 그렇지 않은 경우는 Golden Section Search 방법이 local minimum을 찾는데 더 일반적으로 사용될 수 있다는 생각이 들었다.

## 2. Appendix A : Code

Code는 Python으로 작성했으며 제출할 폴더를 열어보면, search.py와 result.py라는 script가 작성되어있다. search.py에는 FS와 GS가 작성되어 있고, result.py를 실행시키면 결과를 받아볼 수 있다. result.py 안에는 과제에 이용한 함수 4개와 결과를 받아보기 위한 함수가 작성되어있으므로 terminal에서 해당 폴더의 디렉토리로 이동한 후에 "python result.py"를 입력하면 실행시킬 수 있다. 가독성을 위해 코드는 보고서와 함께 따로 첨부했다.

## 3. Appendix B : Some Result

```
Function : f_1
----------------------------------

Fibonacci number list :  [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-5, 1.1803444782168184]
[Root finding...] - iter : 2 - interval : [-5, -1.1803444782168184]
[Root finding...] - iter : 3 - interval : [-5, -2.6393110435663627]
[Root finding...] - iter : 4 - interval : [-5, -3.5410334346504557]
[Root finding...] - iter : 5 - interval : [-5, -4.0982776089159065]
[Root finding...] - iter : 6 - interval : [-5, -4.442755825734549]
[Root finding...] - iter : 7 - interval : [-5, -4.655521783181357]
[Root finding...] - iter : 8 - interval : [-5, -4.787234042553192]
[Root finding...] - iter : 9 - interval : [-5, -4.868287740628166]
[Root finding...] - iter : 10 - interval : [-5, -4.9189463019250255]
[Root finding...] - iter : 11 - interval : [-5, -4.949341438703141]
[Root finding...] - iter : 12 - interval : [-5, -4.9696048632218845]
[Root finding...] - iter : 13 - interval : [-5, -4.979736575481256]
----------------------------------

End point : [-5, -4.979736575481256]

Time :  0.0010089874267578125
Function : f_2
----------------------------------

Fibonacci number list :  [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-5, 1.1803444782168184]
[Root finding...] - iter : 2 - interval : [-5, -1.1803444782168184]
[Root finding...] - iter : 3 - interval : [-5, -2.6393110435663627]
[Root finding...] - iter : 4 - interval : [-5, -3.5410334346504557]
[Root finding...] - iter : 5 - interval : [-5, -4.0982776089159065]
[Root finding...] - iter : 6 - interval : [-5, -4.442755825734549]
[Root finding...] - iter : 7 - interval : [-5, -4.655521783181357]
[Root finding...] - iter : 8 - interval : [-5, -4.787234042553192]
[Root finding...] - iter : 9 - interval : [-5, -4.868287740628166]
[Root finding...] - iter : 10 - interval : [-5, -4.9189463019250255]
[Root finding...] - iter : 11 - interval : [-5, -4.949341438703141]
[Root finding...] - iter : 12 - interval : [-5, -4.9696048632218845]
[Root finding...] - iter : 13 - interval : [-5, -4.979736575481256]
----------------------------------

End point : [-5, -4.979736575481256]
```

```
Function : f_3
-----------------------------------

Fibonacci number list :  [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
-----------------------------------

Start point : [-5, 5]

C:\Users\sanha\NOPT2023\assignment2\result.py:8: RuntimeWarning: invalid value encountered in scalar power
  f3 = lambda x: np.exp(x) - x**np.exp(1) + 3*x**3 -2*x -5  # x = 1.249
[Root finding...] - iter : 1 - interval : [-5, 1.1803444782168184]
[Root finding...] - iter : 2 - interval : [-5, -1.1803444782168184]
[Root finding...] - iter : 3 - interval : [-5, -2.6393110435663627]
[Root finding...] - iter : 4 - interval : [-5, -3.5410334346504557]
[Root finding...] - iter : 5 - interval : [-5, -4.0982776089159065]
[Root finding...] - iter : 6 - interval : [-5, -4.442755825734549]
[Root finding...] - iter : 7 - interval : [-5, -4.655521783181357]
[Root finding...] - iter : 8 - interval : [-5, -4.787234042553192]
[Root finding...] - iter : 9 - interval : [-5, -4.868287740628166]
[Root finding...] - iter : 10 - interval : [-5, -4.9189463019250255]
[Root finding...] - iter : 11 - interval : [-5, -4.949341438703141]
[Root finding...] - iter : 12 - interval : [-5, -4.9696048632218845]
[Root finding...] - iter : 13 - interval : [-5, -4.979736575481256]
-----------------------------------

End point : [-5, -4.979736575481256]

Time :  0.0019998550415039062
Function : f_4
-----------------------------------

Fibonacci number list :  [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
-----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-1.1803444782168184, 5]
[Root finding...] - iter : 2 - interval : [-1.1803444782168184, 1.1803444782168189]
[Root finding...] - iter : 3 - interval : [-0.27864896170105247, 1.1803444782168189]
[Root finding...] - iter : 4 - interval : [-0.27864896170105247, 0.2786489617010528]
[Root finding...] - iter : 5 - interval : [-0.06579211595719281, 0.2786489617010528]
[Root finding...] - iter : 6 - interval : [-0.06579211595719281, 0.06579211595719314]
[Root finding...] - iter : 7 - interval : [-0.015550863771699987, 0.06579211595719314]
[Root finding...] - iter : 8 - interval : [-0.015550863771699987, 0.015550863771700334]
[Root finding...] - iter : 9 - interval : [-0.003702586612309388, 0.015550863771700334]
[Root finding...] - iter : 10 - interval : [-0.003702586612309388, 0.003702586612309735]
[Root finding...] - iter : 11 - interval : [-0.0009256466530772169, 0.003702586612309735]
[Root finding...] - iter : 12 - interval : [-0.0009256466530772169, 0.0009256466530775638]
[Root finding...] - iter : 13 - interval : [-0.0003085488843589567, 0.0009256466530775638]
-----------------------------------

End point : [-0.0003085488843589567, 0.0009256466530775638]
```

```
-----------------------------------

Golden section method
-----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-5, -1.1799999999999997]
[Root finding...] - iter : 2 - interval : [-2.63924, -1.1799999999999997]
[Root finding...] - iter : 3 - interval : [-2.63924, -2.0818103199999998]
[Root finding...] - iter : 4 - interval : [-2.29474845776, -2.0818103199999998]
[Root finding...] - iter : 5 - interval : [-2.29474845776, -2.2134060891356797]
[Root finding...] - iter : 6 - interval : [-2.24447887395017, -2.2134060891356797]
[Root finding...] - iter : 7 - interval : [-2.24447887395017, -2.232609070151035]
[Root finding...] - iter : 8 - interval : [-2.2371433352023042, -2.232609070151035]
[Root finding...] - iter : 9 - interval : [-2.2371433352023042, -2.235411245952719]
[Root finding...] - iter : 10 - interval : [-2.2360729040460607, -2.235411245952719]
[Root finding...] - iter : 11 - interval : [-2.2360729040460607, -2.2358201506544044]
[Root finding...] - iter : 12 - interval : [-2.2359167024500173, -2.2358201506544044]
[Root finding...] - iter : 13 - interval : [-2.2359167024500173, -2.235879819664093]
[Root finding...] - iter : 14 - interval : [-2.235893908888316, -2.235879819664093]
[Root finding...] - iter : 15 - interval : [-2.235893908888316, -2.235888526804663]
-----------------------------------

End point : [-2.235893908888316, -2.235888526804663]

Time :  0.0009968280792236328
Function : f_2
-----------------------------------

Golden section method
-----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-5, -1.1799999999999997]
[Root finding...] - iter : 2 - interval : [-2.63924, -1.1799999999999997]
[Root finding...] - iter : 3 - interval : [-2.63924, -2.0818103199999998]
[Root finding...] - iter : 4 - interval : [-2.29474845776, -2.0818103199999998]
[Root finding...] - iter : 5 - interval : [-2.29474845776, -2.2134060891356797]
[Root finding...] - iter : 6 - interval : [-2.24447887395017, -2.2134060891356797]
[Root finding...] - iter : 7 - interval : [-2.24447887395017, -2.232609070151035]
[Root finding...] - iter : 8 - interval : [-2.2371433352023042, -2.232609070151035]
[Root finding...] - iter : 9 - interval : [-2.2371433352023042, -2.235411245952719]
[Root finding...] - iter : 10 - interval : [-2.2360729040460607, -2.235411245952719]
[Root finding...] - iter : 11 - interval : [-2.2360729040460607, -2.2358201506544044]
[Root finding...] - iter : 12 - interval : [-2.2359167024500173, -2.2358201506544044]
[Root finding...] - iter : 13 - interval : [-2.2359167024500173, -2.235879819664093]
[Root finding...] - iter : 14 - interval : [-2.235893908888316, -2.235879819664093]
[Root finding...] - iter : 15 - interval : [-2.235893908888316, -2.235888526804663]
-----------------------------------

End point : [-2.235893908888316, -2.235888526804663]
```

```
-----------------------------------

Golden section method
-----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-5, -1.1799999999999997]
[Root finding...] - iter : 2 - interval : [-2.63924, -1.1799999999999997]
[Root finding...] - iter : 3 - interval : [-2.63924, -2.0818103199999998]
[Root finding...] - iter : 4 - interval : [-2.29474845776, -2.0818103199999998]
[Root finding...] - iter : 5 - interval : [-2.29474845776, -2.2134060891356797]
[Root finding...] - iter : 6 - interval : [-2.24447887395017, -2.2134060891356797]
[Root finding...] - iter : 7 - interval : [-2.24447887395017, -2.232609070151035]
[Root finding...] - iter : 8 - interval : [-2.2371433352023042, -2.232609070151035]
[Root finding...] - iter : 9 - interval : [-2.2371433352023042, -2.235411245952719]
[Root finding...] - iter : 10 - interval : [-2.2360729040460607, -2.235411245952719]
[Root finding...] - iter : 11 - interval : [-2.2360729040460607, -2.2358201506544044]
[Root finding...] - iter : 12 - interval : [-2.2359167024500173, -2.2358201506544044]
[Root finding...] - iter : 13 - interval : [-2.2359167024500173, -2.235879819664093]
[Root finding...] - iter : 14 - interval : [-2.235893908888316, -2.235879819664093]
[Root finding...] - iter : 15 - interval : [-2.235893908888316, -2.235888526804663]
-----------------------------------

End point : [-2.235893908888316, -2.235888526804663]

Time :  0.0010008811950683594
Function : f_4
-----------------------------------

Golden section method
-----------------------------------

Start point : [-5, 5]

[Root finding...] - iter : 1 - interval : [-5, -1.1799999999999997]
[Root finding...] - iter : 2 - interval : [-2.63924, -1.1799999999999997]
[Root finding...] - iter : 3 - interval : [-2.63924, -2.0818103199999998]
[Root finding...] - iter : 4 - interval : [-2.29474845776, -2.0818103199999998]
[Root finding...] - iter : 5 - interval : [-2.29474845776, -2.2134060891356797]
[Root finding...] - iter : 6 - interval : [-2.24447887395017, -2.2134060891356797]
[Root finding...] - iter : 7 - interval : [-2.24447887395017, -2.232609070151035]
[Root finding...] - iter : 8 - interval : [-2.2371433352023042, -2.232609070151035]
[Root finding...] - iter : 9 - interval : [-2.2371433352023042, -2.235411245952719]
[Root finding...] - iter : 10 - interval : [-2.2360729040460607, -2.235411245952719]
[Root finding...] - iter : 11 - interval : [-2.2360729040460607, -2.2358201506544044]
[Root finding...] - iter : 12 - interval : [-2.2359167024500173, -2.2358201506544044]
[Root finding...] - iter : 13 - interval : [-2.2359167024500173, -2.235879819664093]
[Root finding...] - iter : 14 - interval : [-2.235893908888316, -2.235879819664093]
[Root finding...] - iter : 15 - interval : [-2.235893908888316, -2.235888526804663]
-----------------------------------

End point : [-2.235893908888316, -2.235888526804663]

Time :  0.0009999275207519531
```