Root-finding 과제 보고서

2019055078 신채영

* zbrak를 사용해서 bracket을 정하고
* NR파일에있는 root finding method들을 콜한다
  + Bisection
    - 가장 간단하지만 느림
    - 구간의 중점에서 시작해서 중점의 y부호를 보고 답이 있을 수 없는 구간을 배제시키면서 구간을 반 씩 좁혀가는 방식
  + Linear interpolation
    - Bisection을 개선한 방법
    - 구간 양끝의 점을 이어 x축과의 접점의 y부호를 확인하고 답이 있을 수 없는 구간을 배제하고 그 점과 반대쪽 끝으로 다시 선을 이어 반복하는 방식
  + Secant
    - 뉴턴이 기본적인 발상이지만 순간 기울기가 아닌 평균 기울기를 이용
    - 구간의 양 끝(p0,p1)을 연결한 직선과 x축의 접점을 p2라고 하고

다시 p2와 p1을 연결한 직선과 x축의 접점을 p3이라고 하고

이대로 반복…

* + - Linear interpolation과 복잡도는 비슷하나 좀 더 빠르고
    - 뉴턴보다 안정적
  + Newton-raphson
    - 시작점의 순간 기울기와 x축의 접점을 점 a라 하고 a의 x값에 대응하는 함수 위의 점에서 다시 순간 기울기를 구하면서 반복해 답에 접근하는 방식
    - 빠르고 quadratic convergence이다
    - 불안정성: 답에서 멀어지거나 잡을 찾지 못하는 cycling, wandering같은 문제들이 있다.
    - 도함수를 계산해야 하기 때문에 계산이 복잡하다
  + Newton with bracketing (safe)
    - 뉴턴의 불안정성을 보안하기 위해 bracket의 장점인 안정성과 결합한 방식
    - 뉴턴을 쓰는 대신 x가 범위를 벗어나거나 너무 작으면 bisection을 사용
  + Muller
    - 직선을 사용하지 않고 곡선으로 root에 접근하는 방식
    - 3개의 점으로 2차함수를 구하고 첫번째 점을 빼고 계속 해서 3개의 점으로 곡선을 만들고 root를 구한다
* Convergence speed:
  + 코드에서 root finding 함수를 콜하면서 time\_t start와 end를 선언해서 걸린 시간을 구해서 각 걸리는 시간을 구한다. 모두 아주 작은 수라서 보이진 않지만 걸리는 시간은

Bisection > linear interpolation > secant > newton > safe 이다.

* 식 3개:
  + Bessel 의 root를 뉴턴과 safe로 구할 때와 같은 방법으로 도함수를 구하고 safe를 호출한다.