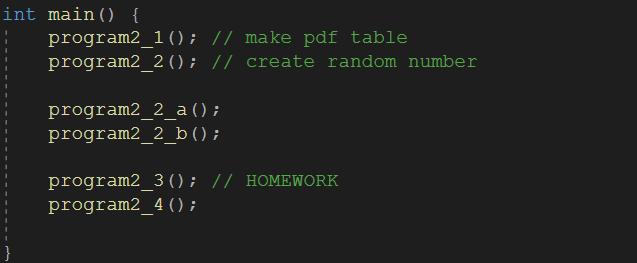
**고급소프트웨어실습 5주차 보고서**

**20191657 최세은**

1. 프로그램 구동 방법



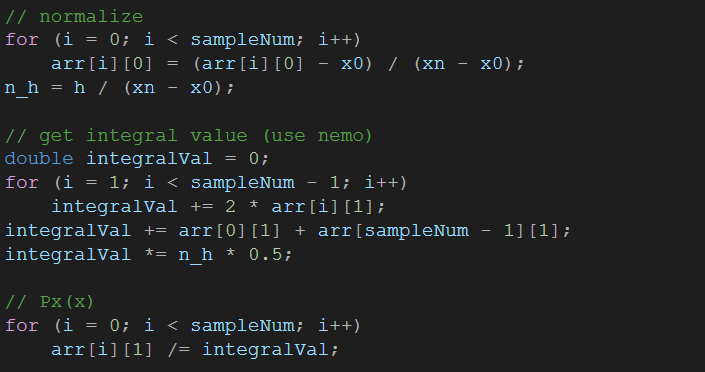
해당 순서로 함수들이 실행된다.

program2\_1은 사용자 입력 없이 실행되고, program2\_2, program2\_2\_a, program2\_2\_b 에서는 만들고 싶은 난수의 개수를 입력해준다. 이후 program2\_3에서는 히스토그램 데이터의 구간 개수를 입력하고 program2\_4 에서는 난수의 개수와 람다값을 입력해준다.

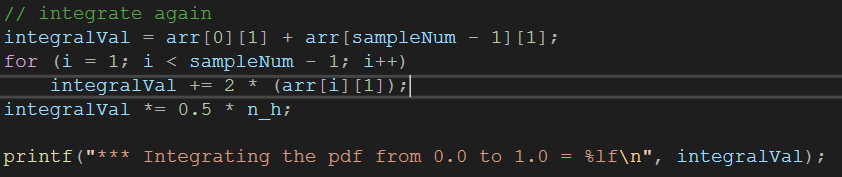
2. 실습

2-1. program2\_1

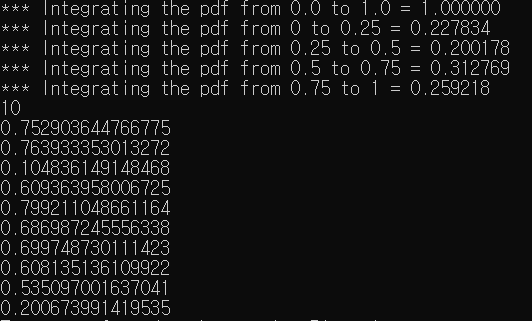
sampling\_table.txt 파일에 저장되어 있는 함수 y = f(x)의 샘플 데이터를 입력 받아, 그에 해당하는 확률 밀도 함수에 대한 샘플 데이터를 pdf\_table.txt에 저장해주는 함수이다.



이 때 x 구간을 정규화하고, 확률 밀도 함수 P(x)를 계산한다.

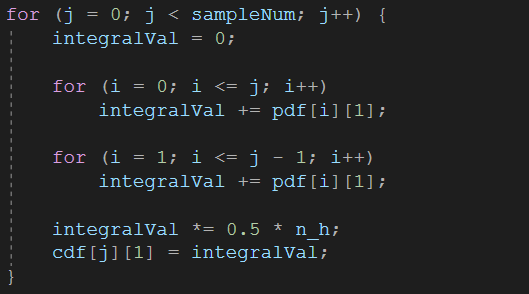


이후 전체 구간에 대해 다시 적분해서 값을 확인하고, 여러 구간으로 나누어 부분 적분 하여 값을 또 확인한다.

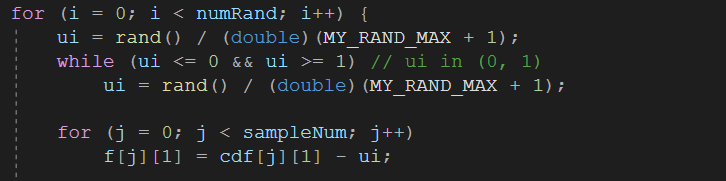


그러면 위와 같이 적분 결과가 콘솔창에 뜬다.

2-2. program2\_2



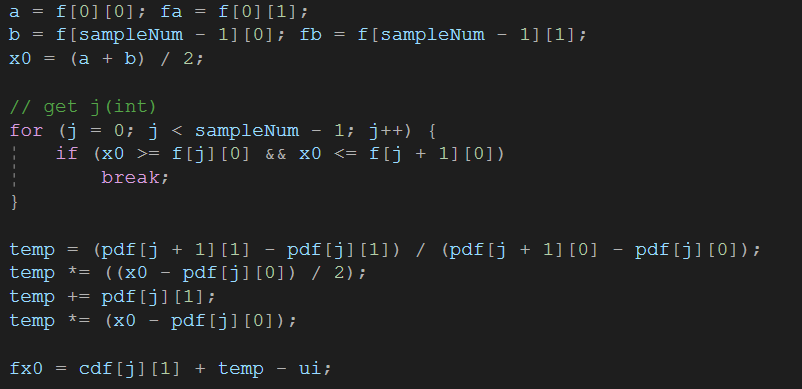
우선 program2\_1 함수에서 만든 pdf\_table 파일을 불러와 pdf 배열에 값을 알맞게 넣어준다. 이후 이 값을 이용해 적분 값을 구해 cdf 배열에 저장한다.



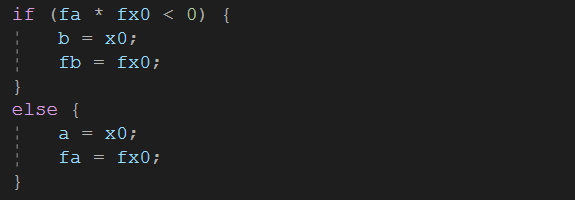
이후 콘솔 창에 만들고 싶은 난수의 개수 numRand를 입력받고, 이 수로 반복문을 돌리며 유사 난수를 생성한다. 난수를 생성하기 위해 우선 rand 함수로 0과 1사이의 난수를 ui에 받는다. 이후 이와 cdf 값을 이용하여



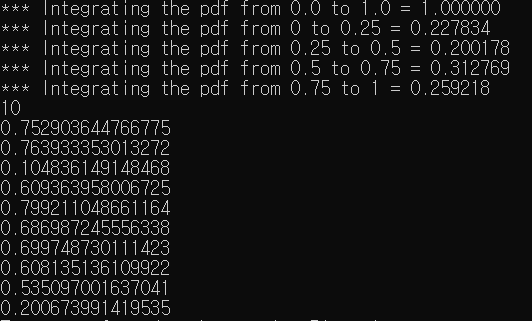
이 함수 값을 구해 f 배열에 저장한다.



이후 bisection 방법으로 난수를 계산하게 되는데, 우선 초기값 a(=f[0][0]), b(=f[xn][0])을 정하고 중간값 x0을 가지는 구간 [j, j + 1]을 구한 다음 이 j를 이용해 f(x)를 구한다.



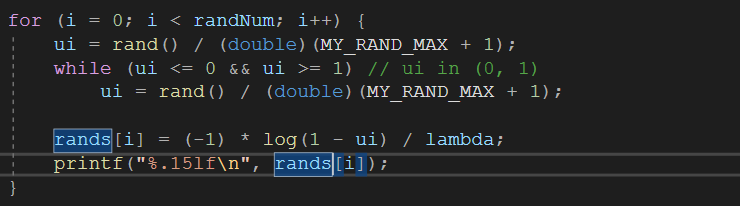
그리고 그리고 j와 j + 1로 나누어진 두 구간에서 근이 존재하는 구간을 찾아 다음 구간으로 지정하고 이를 반복문으로 계속 반복한다. 이는 저번 주 실습에서 사용했던 bisection 방법이다.



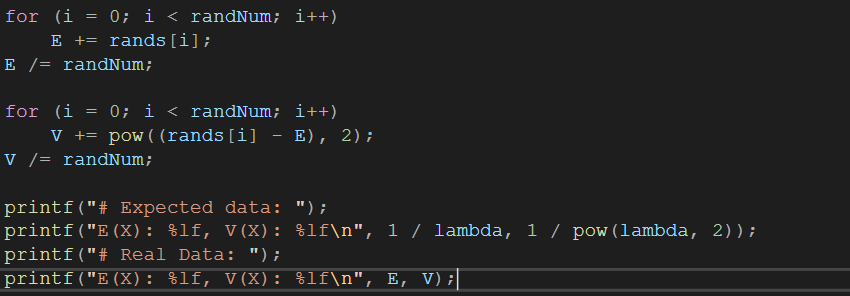
이러면 위와 같이 사용자가 입력한 수만큼 난수가 생성된다.

3. 과제

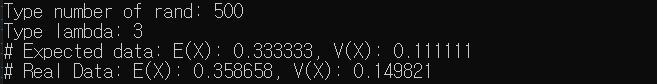
3-1. program2\_4

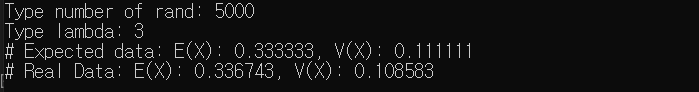


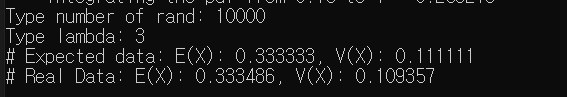
우선 생성할 난수의 개수와 람다를 입력 받았다. 그리고 rand 함수로 생성한 수 ui로 지수 분포를 따르는 난수들을 얻고 출력했다.

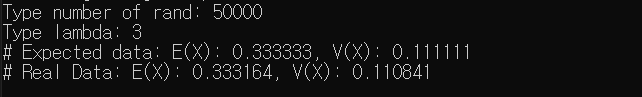


그리고 입력한 람다와 rands에 저장된 난수들로 기대한 평균 및 분산값과 실제 평균 및 분산값을 계산해 출력했다.



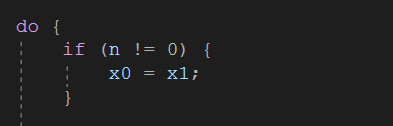






결과를 확인해보면 예상했던 대로 난수가 많아질수록 정확한 값이 도출되었고, 난수의 개수가 10000개일 때나 50000개일 때나 정확도의 차이가 별로 없는 것으로 보아 10000개일 때부터 유의미한 결과가 도출됨을 알 수 있었다.

3-2. program2\_2\_a



우선 program2\_2 코드에서 n이 0에서 더해지지 못하는 문제점을 발견해서 코드를 수정했다. n 값이 변하지 않아도 다른 조건을 만족하면 조건문이 종료되지만 실행 시간의 효율성을 위해 수정했다.

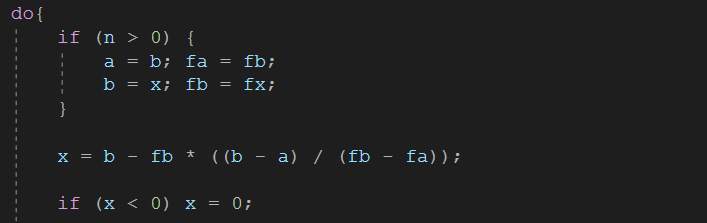
이 그래프는 program2\_2\_a를 실행한 후 program2\_3으로 만들어진 histogram.txt을 분석한 그래프이다.

이 그래프는 pdf\_table.txt 파일로 만든 그래프이다. 위 histogram 그래프의 표본이 비교적 적으나, 전체적인 모양이 거의 같음을 알 수 있다. 을 통하여 Xi를 찾기 때문에 Fx(x)의 변화량이 가장 큰 지점에서 난수가 많이 생기게 된다. 즉 확률 밀도 함수의 함숫값이 가장 큰 지점에서 난수가 가장 많이 생성되어야 하므로 이 두 그래프의 모양이 비슷한 것은 유효한 program2\_2\_a를 만들었다는 결론이 된다.

3-3. program2\_2\_b

(1) secant



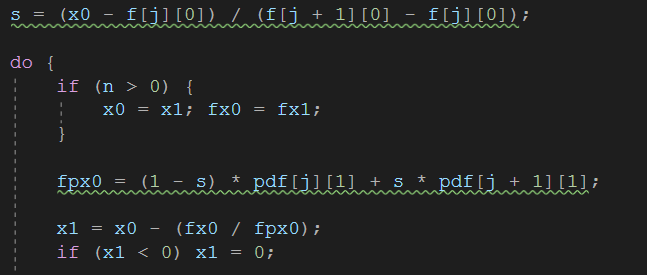


bisection이 아닌 secant를 이용해서 난수를 구하는 프로그램이다. 따라서 secant를 사용하는 부분을 제외하면 program2\_2\_a와 거의 같게 코드가 짜여졌다. 초기값은 a, b 즉 x0, xn으로 주었다.

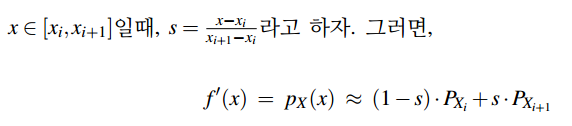
(2) newton-ralpse



sampleNum개로 균등하게 데이터 구간이 나뉘었기 때문에 초기값은 x0, 구간의 중간값으로 정했다.

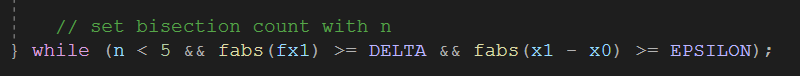


이후 밑의 공식을 이용해 난수를 생성해간다.



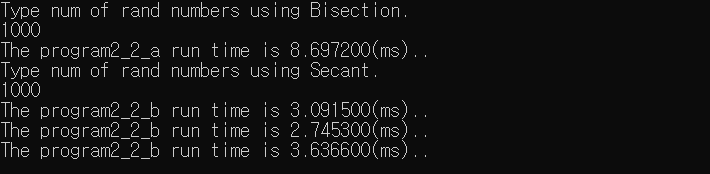
(3) newton-ralpse with bisection

자료의 설명대로 [x0, xn] 구간을 초기값으로 하여 bisection을 n회 정도 반복한 후, 중점을 newton-ralpse 알고리즘을 사용할 때의 초기값으로 사용하는 방법이다.



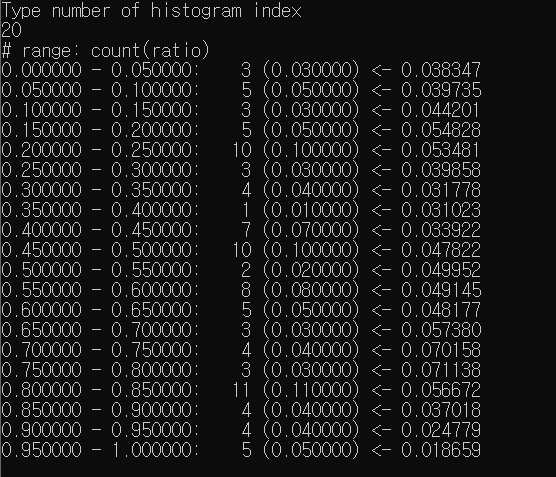
이 때 반복 횟수 n은 bisection을 수행하는 반복문에서 종료 조건으로 결정할 수 있다. (캡쳐 속 숫자 5)

이렇게 program2\_2\_a에서의 한가지 방법, program2\_2\_b에서의 세가지 방법으로 난수를 생성한 결과는 다음과 같다.



차례대로 program2\_2\_a, program2\_2\_b의 결과이다. bisection > newton with bisection > secant > newton 순으로 실행 시간이 오래 걸린다. 이 때 newton with bisection에서 bisection 횟수를 여러 번 조절해보았으나, newton 알고리즘만 사용할 때보다 항상 오래 걸렸다.

3-4. program2\_3



program2\_2 등 프로그램을 이용하여 난수를 생성한 후, 이를 분석해 histogram.txt를 생성하는 프로그램이다. 사용자에게 히스토그램의 구간 개수(index)를 입력 받고, 이와 미리 다른 프로그램으로 생성해둔 random\_event\_table.txt(난수들)을 통해 구간 별 난수의 개수 및 비율, 그리고 각 구간별 재적분값을 출력한다.

이를 통해 위 그래프가 생성되며, 얘기했듯 원래 그래프로 만든 확률 변수와의 비교를 통해 난수가 잘 생성되었는 지 분석할 수 있다.