**고급 소프트웨어실험 5주차 과제**

**프로그램 구동 방법**

각 함수의 기능은 다음과 같다.

* Program2\_1: pdf table을 생성한다.
* Program2\_2: 기본 Bisection 방식을 사용하여 난수를 생성한다.
* Program2\_1\_a: 람다와 시행횟수를 입력받아 지수분포를 따르는 난수를 생성하고 이의 기댓값과 분산을 계산하여 이론값과 비교한다.
* Program2\_2\_a: 조금 더 효율적인 Bisection 방식을 사용하여 난수를 생성한다.
* Program2\_2\_b: Secant 방식을 사용하여 난수를 생성한다.
* Program2\_2\_c: Newton-Rapson 방식을 사용하여 난수를 생성한다.
* Program2\_3: pdf table과 생성된 난수를 비교하여 난수가 잘 생성되었는지 확인 할 수 있는 histogram을 생성한다. Histogram 은 구간, 구간의 확률분포값, 구간에서 생성된 난수의 전체 생성된 난수에 대한 비율을 저장한다.

프로그램을 main 함수에서 실행하고자 하는 코드의 주석을 해제하는 방식으로 사용하면 된다.

**과제 문제 2-1**

1. **수행 결과**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| λ | 시행 횟수 | E(x) (이론) | E(x) | V(x) (이론) | V(x) |
| 0.5 | 1000 | 2 | 2.038890 | 4 | 3.684831 |
| 10000 | 1.969773 | 3.895675 |
| 100000 | 2.000023 | 3.968461 |
| 1 | 1000 | 1 | 1.016559 | 1 | 0.985595 |
| 10000 | 1.003510 | 1.011638 |
| 100000 | 0.997039 | 0.990795 |
| 1.5 | 1000 | 0.666667 | 0.648183 | 0.444444 | 0.413373 |
| 10000 | 0.666771 | 0.454454 |
| 100000 | 0.668782 | 0.445463 |

1. **분석**

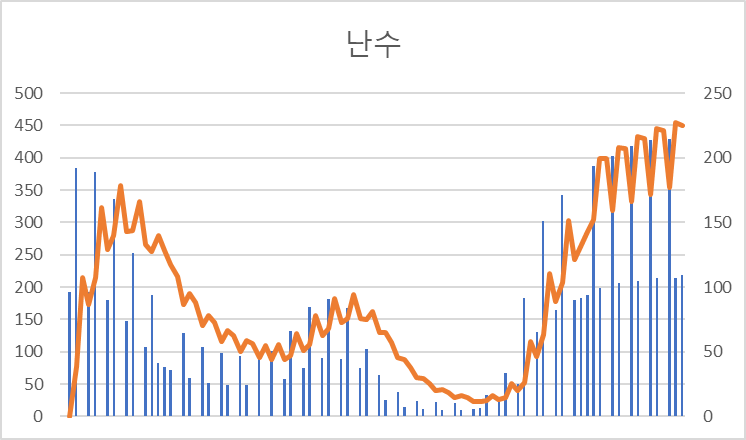
* 수행 결과를 통해 이론값과 실제 발생한 난수를 통해 계산한 값이 유사함을 확인할 수 있었으며, 시행횟수가 커질수록 이론값과 실험값 사이의 오차가 작아지는 경향을 보임을 확인할 수 있었다.

**과제 문제 2-2**

모든 실험에서 생성된 난수의 개수는 1000000개로 동일하다. 코드가 효율적으로 실행될 수 있도록, 실습시간에 작성한 코드에서 구간별 적분 값을 미리 계산하여 저장해두는 Dynamic Programming 방식을 사용하였다.

1. Bisection Method

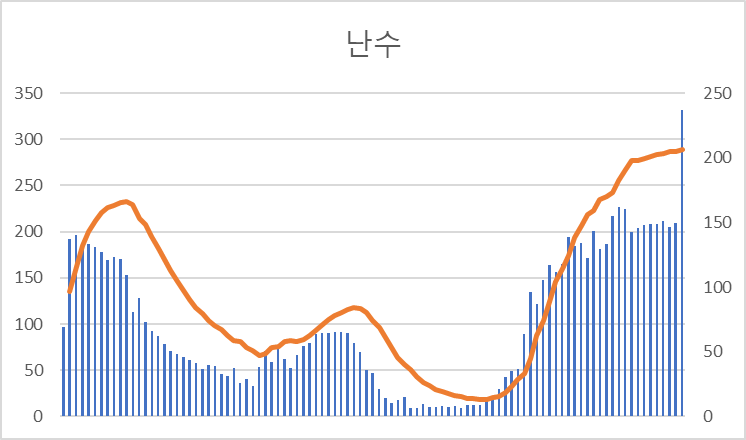
* 수행결과



소요시간: 17676.033020(ms)

1. Secant Method

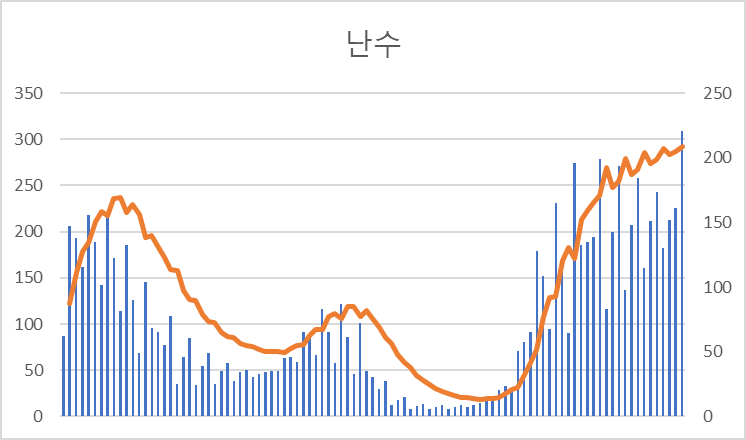
* 수행결과



소요시간: 17165.536880(ms).

1. Newton-Rapson Method

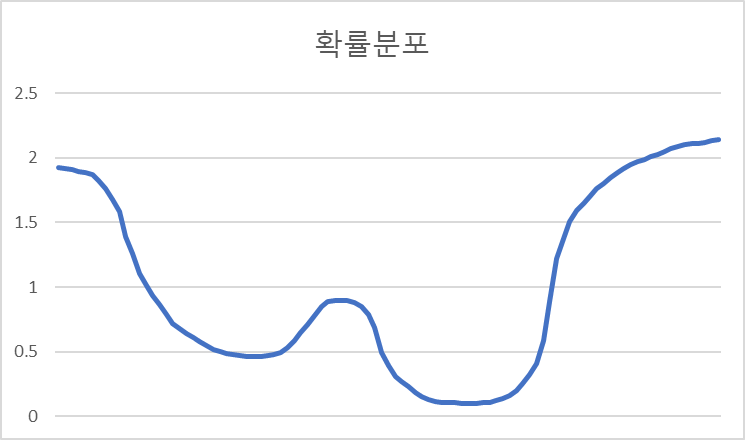
* 수행결과



소요시간: 17144.119263(ms)

1. 분석

* 그래프의 파란 막대는 해당 구간에서 발생한 난수의 개수의 전체에 대한 비율을, 주황색은 난수 발생의 추세를 보기 위해 이를 지수 평활법을 사용하여 평탄화 한 값을 나타낸다. 이 때 확률분포 그래프는 아래와 같다.



각 방법에서 만들어진 결과와 확률분포 그래프를 비교해보면, 그 모양이 대략적으로 일치함을 알 수 있고, 따라서 난수가 확률분포대로 잘 생성되고 있음을 확인할 수 있다. 다만, Bisection 방법에서와 Newton-Rapson 방법에서의 경우 그래프에 노이즈가 낀 듯 들쭉날쭉한 부분을 확인할 수 있는데, 이는 두 방법에서 근을 구하는 과정에서 특정한 영역에 있는 난수가 다른 영역에서의 난수보다 발생하기 어렵기 때문이다. 이러한 오차를 줄이기 위해서는 각 방법에서 반복문의 종료조건을 수정하여 실제 근과 더 오차가 적은 근을 찾게하면 된다. 하지만, 이렇게 할 경우 한 근을 찾는데에 드는 비용이 커지므로 효율적이지 못하다.

세 방법에서 소요되는 시간은 비슷했다. 하지만, Secant 방법을 사용할 때의 정확도가 나머지 두 방법에서보다 크므로, 이 상황에서 Secant 방법이 타 방법보다 조금 더 유용하다고 볼 수 있다.