**고급소프트웨어실습1**

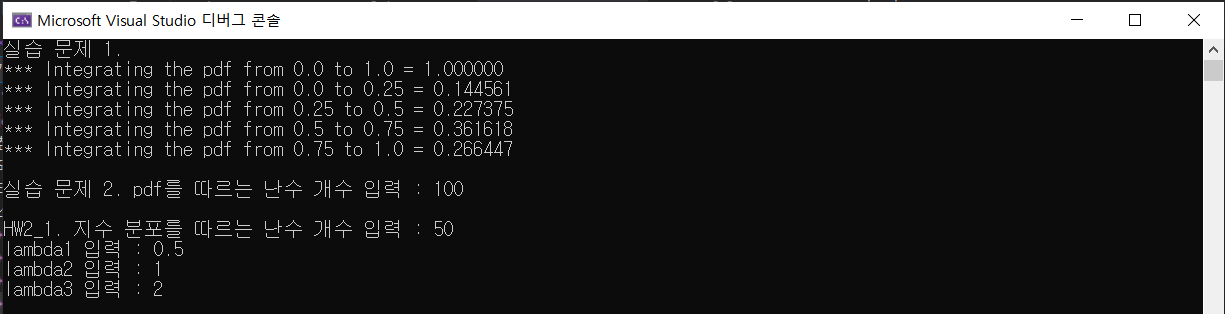
**5주차 과제**

**20161663 허재성**

**숙제 2-1**

교재의 지수 분포 함수와 inversion, 그리고 inversion을 위해 Newton-Rapson 방법을 이용하여 지수 분포

를 따르는 난수를 생성하였다. 프로그램을 실행시킨 뒤 실습 2에서 생성할 난수의 개수를 입력하면, 실습 2의 결과가 random\_event\_table.txt에 저장되고 숙제 2-1의 지수 분포를 따르는 난수를 구하기 위한 입력을 받는다.



생성할 난수의 개수와 서로 다른 세 개의 지수 분포에서의 난수를 구하기 위해 원하는lambda()값을 입력하면 된다. 아래는 난수의 개수를 50개로, lambda 값을 각각 0.5, 1, 2로 했을 때의 결과이다. 난수의 값은 hw2\_1result.txt에서 확인할 수 있다.

|  |  |  |  |
| --- | --- | --- | --- |
| **lambda** | **0.5** | **1** | **2** |
| 1 | 1.462625 | 0.731313 | 0.365656 |
| 2 | 0.074936 | 0.037468 | 0.018734 |
| 3 | 0.094933 | 0.047467 | 0.023733 |
| 4 | 4.313584 | 2.156792 | 1.078396 |
| 5 | 1.813719 | 0.906859 | 0.45343 |
| 6 | 3.28351 | 1.641755 | 0.820877 |
| 7 | 1.685927 | 0.842963 | 0.421482 |
| 8 | 0.728922 | 0.364461 | 0.182231 |
| 9 | 4.638839 | 2.31942 | 1.15971 |
| 10 | 1.64411 | 0.822055 | 0.411027 |
| 11 | 0.360241 | 0.180121 | 0.09006 |
| 12 | 0.828648 | 0.414324 | 0.207162 |
| 13 | 0.357466 | 0.178733 | 0.089366 |
| 14 | 0.490069 | 0.245034 | 0.122517 |
| 15 | 0.617576 | 0.308788 | 0.154394 |
| 16 | 1.942161 | 0.97108 | 0.48554 |
| 17 | 0.366023 | 0.183012 | 0.091506 |
| 18 | 3.747209 | 1.873604 | 0.936802 |
| 19 | 4.427001 | 2.213501 | 1.10675 |
| 20 | 4.920319 | 2.46016 | 1.23008 |
| 21 | 2.201502 | 1.100751 | 0.550375 |
| 22 | 1.607507 | 0.803754 | 0.401877 |
| 23 | 0.249016 | 0.124508 | 0.062254 |
| 24 | 0.754126 | 0.377063 | 0.188531 |
| 25 | 4.574101 | 2.28705 | 1.143525 |
| 26 | 1.696305 | 0.848153 | 0.424076 |
| 27 | 4.931785 | 2.465892 | 1.232946 |
| 28 | 0.920316 | 0.460158 | 0.230079 |
| 29 | 0.81787 | 0.408935 | 0.204467 |
| 30 | 1.905586 | 0.952793 | 0.476396 |
| 31 | 0.575629 | 0.287814 | 0.143907 |
| 32 | 1.019943 | 0.509971 | 0.254986 |
| 33 | 3.999789 | 1.999894 | 0.999947 |
| 34 | 7.712294 | 3.856147 | 1.928074 |
| 35 | 3.917622 | 1.958811 | 0.979405 |
| 36 | 3.180308 | 1.590154 | 0.795077 |
| 37 | 0.030816 | 0.015408 | 0.007704 |
| 38 | 1.293915 | 0.646957 | 0.323479 |
| 39 | 1.861136 | 0.930568 | 0.465284 |
| 40 | 1.708172 | 0.854086 | 0.427043 |
| 41 | 1.077792 | 0.538896 | 0.269448 |
| 42 | 2.350014 | 1.175007 | 0.587504 |
| 43 | 3.474193 | 1.737097 | 0.868548 |
| 44 | 7.271209 | 3.635604 | 1.817802 |
| 45 | 0.274689 | 0.137345 | 0.068672 |
| 46 | 1.88637 | 0.943185 | 0.471593 |
| 47 | 1.60778 | 0.80389 | 0.401945 |
| 48 | 0.946206 | 0.473103 | 0.236552 |
| 49 | 0.011508 | 0.005754 | 0.002877 |
| 50 | 0.337929 | 0.168965 | 0.084482 |

한편 지수 분포의 평균과 분산은 이론적으로 다음과 같다.

이를 토대로 각각의 lambda에 대하여 이론적인 평균과 분산을 구하여 위의 표의 난수의 평균, 분산과 비교해보면 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| **실제 평균** | 2.039865 | 1.019932 | 0.509966 |
| **이론적 평균** | 2 | 1 | 0.5 |
| **실제 분산** | 3.433647 | 0.858412 | 0.214603 |
| **이론적 분산** | 4 | 1 | 0.25 |
| **오차율(평균)** | 0.019932 | 0.019932 | 0.019932 |
| **오차율(분산)** | 0.141588 | 0.141588 | 0.141588 |

난수의 개수를 충분히 크게 하기 위해 10000으로 설정 후 결과를 확인해 봤다.

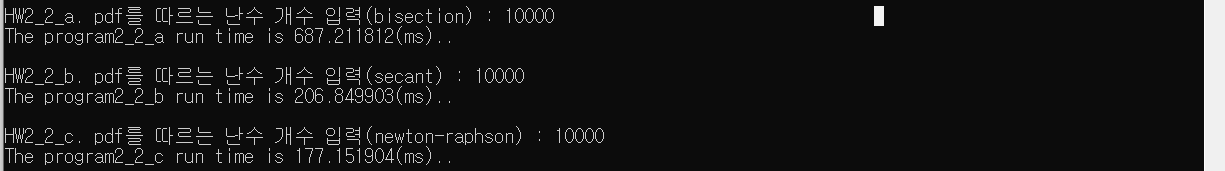
|  |  |  |  |
| --- | --- | --- | --- |
| **lambda** | **0.5** | **1** | **2** |
| 1 | 6.011524 | 3.005762 | 1.502881 |
| 2 | 0.945912 | 0.472956 | 0.236478 |
| 3 | 0.431964 | 0.215982 | 0.107991 |
| 4 | 2.501898 | 1.250949 | 0.625474 |
| 5 | 6.2607 | 3.13035 | 1.565175 |
| 6 | 0.389245 | 0.194623 | 0.097311 |
| 7 | 0.249846 | 0.124923 | 0.062462 |
| 8 | 2.089807 | 1.044903 | 0.522452 |
| 9 | 0.038579 | 0.019289 | 0.009645 |
| 10 | 1.209019 | 0.604509 | 0.302255 |

중략

|  |  |  |  |
| --- | --- | --- | --- |
| 9991 | 2.478999 | 1.239499 | 0.61975 |
| 9992 | 1.751511 | 0.875756 | 0.437878 |
| 9993 | 1.180732 | 0.590366 | 0.295183 |
| 9994 | 0.866318 | 0.433159 | 0.21658 |
| 9995 | 3.724288 | 1.862144 | 0.931072 |
| 9996 | 0.367783 | 0.183891 | 0.091946 |
| 9997 | 9.413635 | 4.706818 | 2.353409 |
| 9998 | 0.083891 | 0.041945 | 0.020973 |
| 9999 | 0.300838 | 0.150419 | 0.07521 |
| 10000 | 0.393105 | 0.196552 | 0.098276 |
| **실제 평균** | 1.998676943 | 0.999338473 | 0.49966923 |
| **이론적 평균** | 2 | 1 | 0.5 |
| **실제 분산** | 4.139729414 | 1.03493236 | 0.258733089 |
| **이론적 분산** | 4 | 1 | 0.25 |
| **오차율(평균)** | 0.000661529 | 0.000661527 | 0.00066154 |
| **오차율(분산)** | 0.034932354 | 0.03493236 | 0.034932354 |

생성된 난수의 개수가 10000으로 크게 늘어나자 평균, 분산의 오차율이 크게 줄어들었다. 난수의 개수가 50일 때 평균의 오차율은 약 2%, 분산의 오차율은 14%나 되었지만 난수의 개수를 10000개로 늘린 결과 평균의 오차율은 0.06%, 분산의 오차율은 3.4%로 크게 줄었다.1000개 정도의 난수로도 충분히 지수 분포를 모사할 수 있다는 결론을 도출할 수 있다.

**숙제 2-2**

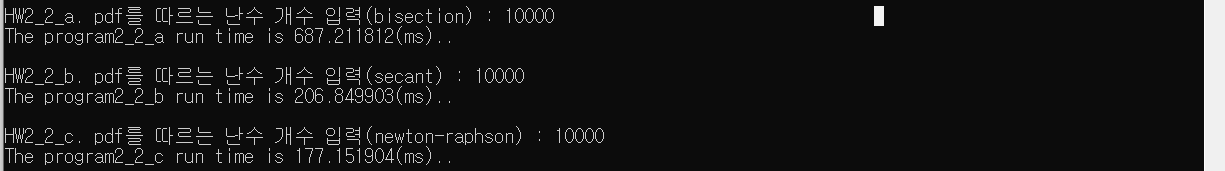
실습에서 생성한 pdf\_table.txt를 이용해 실습과 비슷한 방법으로 pdf를 따르는 난수를 생성한다. 단 시간 측정 코드를 추가하고 실습에선 bisection 방법만을 이용하여 난수를 생성했지만 과제에서는 secant 방법과 Newton-Raphson 방법도 같이 사용하여 세 가지 방법으로 난수를 구해보고 또한 세 방법의 수행 시간도 비교해 본다.

숙제 2-1의 실행이 끝나면 차례대로 bisection, secant, newton-raphson 방법으로 난수를 생성하기 위해 난수 개수를 입력받는다. 세 방법의 시간을 비교하기 위해 같은 개수의 난수를 생성하기로 했으며 충분히 많은 수를 생성하기 위해 개수를 10000개로 하였다. 생성된 난수는 차례대로 HW2\_2\_a\_result.txt, HW2\_2\_b\_result.txt, HW2\_2\_c\_result.txt에 저장되었다.

Newton-Raphson, Secant, Bisection 방법 모두 기본적인 틀은 4주차 실습과 과제에서 구현한 것을 바탕으로 하였다. 차이가 있다면 Newton-Raphson과 Secant 방법에서 초기값을 설정하는 방법인데 초기값을 설정하기 위해 Bisection 방법을 사용하였다. Newton-Raphson의 경우 초기값이 근에 가까울수록 근을 찾는 속도가 빨라지는 것을 지난 실습을 통해 알고 있다. 이를 위해 근에 가까운 초기값을 구하기 위해 구간 [0, 1]에 bisection 방법을 10회 적용해 근과 가까운 초기값을 논리적으로 찾아냈다. 이 때 Bisection 방법으로 구간을 절반의 크기로 줄여 나가며 10번째 구간의 중앙값을 초기값으로 설정하였다.

Newton-Raphson 방법을 구현하기 위해 Fx(x) – U 의 도함수가 필요한데 도함수가 확률 밀도 함수 Px(x)임은 이미 알고 있다. Px(x)는 sampling된 결과만을 가지고 있을 뿐, 수식으로 구현되어 있지 않지만 강의 자료의 선형 보간 방법을 구현하여 Px(x)를 반환하는 pdf 함수를 구현하였다.

Secant 방법의 경우 두 개의 초기값이 필요한데 이를 근이 존재하는 구간의 양 끝 점의 x 좌표로 설정한다. 이를 위해 Newton-Raphson 방법과 마찬가지로 Bisection 방법을 사용했다. Bisection 방법으로 구간 [0, 1]을 절반씩 10번 줄여가며 근이 존재하는 구간을 찾은 뒤에 해당 구간의 양 끝점의 x 좌표를 Secant 방법을 위한 초기값으로 하였다.



10000개의 난수를 생성하는 데 Bisection 방법의 경우 687ms가 소요되었다. 여러 번 실험해봐도 600ms 아래로 나온 적이 없었다. 중간값의 정리를 이용하여 근이 존재한다는 것만 아는 구간에서 구간을 절반씩 줄여가며 근을 찾아서 시간이 가장 오래 걸린다.

Secant 방법의 경우 206ms가 소요되었다. Bisection 방법보다 훨씬 빠르게 근을 찾을 수 있었다. 아래의 Newton-Raphson보다는 조금 느리지만 큰 차이는 아니었다. 4주차에서 확인한 Secant 방법이 Newton-Raphson 방법보다 근을 찾는 속도가 조금은 느리지만 그래도 빠르게 근에 수렴한다는 것을 확인했다. Newton-Raphson 방법은 177ms가 소요되었으며 가장 빠르게 근을 찾을 수 있었다.

위에서 근을 찾는다는 것은 임의의 난수 U에 대하여 U = Fx(X)를 만족하는 X를 찾는다, 즉 확률 분포를 따르는 임의의 난수를 생성한다는 것을 의미한다.

pdf\_table.txt에 저장된 확률 밀도 함수의 sampling 결과를 이용해 10000개의 난수를 생성해 random\_event\_table.txt에 저장하였다. 구간 [0, 1]을 99개의 구간으로 나누었을 때(pdf를 생성하기 위해 나눈 구간의 개수와 동일하다.) 각 구간에 존재하는 난수의 개수를 구해서 histogram.txt에 저장하였다. 이를 이용해 히스토그램을 그려보면 다음과 같다.

구간은 h = 1/99일 때, [0, h], [h, 2h], … , [98h, 99h] 총 9개이며, 히스토그램의 x축은 구간[0, h]을 1이라 할 때, 1 ~ 99의 구간을 나타낸다. 히스토그램을 pdf\_table.txt에 저장된 pdf를 샘플링한 것을 그래프로 나타낸 것과 비교해 본다.

임의로 생성한 pdf(pdf\_table.txt)와 pdf로 생성한 10000개의 난수의 히스토그램이 어느 정도 차이는 있지만 전체적으로 일치하는 것을 확인할 수 있다. 즉 생성한 난수가 임의로 생성한 확률 밀도 함수를 따르는 것을 알 수 있다.

위의 histogram.txt는 Bisection 방법으로 생성한 난수로 생성한 것이다. Bisection 방법으로 생성한 난수가 확률밀도함수에 어느 정도 일치하는 것을 확인했으니 Secant와 Newton-Raphson 방법으로 생성한 난수도 확인해본다. 이를 위해 HW2\_2\_b\_result.txt, HW2\_2\_c\_result.txt에 저장된 난수를 읽어서 histogram\_b.txt, histrogram\_c.txt를 생성 후 히스토그램을 그려본다.

Secant 방법으로 생성한 난수의 히스토그램이다. Outlier가 존재하긴 하지만 pdf와 추세가 거의 동일하다는 것을 확인 가능하다.

마찬가지로 Newton-Raphson 방법으로 생성한 난수의 히스토그램이다. Pdf와 추세가 거의 동일한 것을 확인할 수 있다.